MANCHESTER 1824
The University of Manchester

COMP30040 Report

A report submitted to The University of Manchester for the degree of
Bachelor of Science in Computer Science
in the Faculty of Science and Engineering

Year of submission
2025

Student ID
10826115

School of Computer Science

Contents

Word count: TODO

List of figures

List of tables

Abbreviations and Acronyms

AI Artificial Intelligence

API Application Programming Interface

CD Compact Disc

CDPA Copyright, Designs and Patents Act 1988

CLK Clock (regular pulses used for quadrature rotary encoding)

CNN Convolutional Neural Network

CV Computer Vision

DT Data (offset pulses used for quadrature rotary encoding)

EN Enable

GDPR General Data Protection Regulation

GPIO General-Purpose Input/Output

GPU Graphical Processing Unit

ML Machine Learning

OCR Optical Character Recognition

OS Operating System

PWM Pulse-Width Modulation

REST Representational State Transfer

RPi Raspberry Pi

RPiOS Raspberry Pi OS (formerly Raspbian)

RPM Rotations per Minute

SBC Single-Board Computer

SDK Software Development Kit

SRP Single Responsibility Principle

SSE Server-Sent Events

SW Switch (the live connection of a button circuit)

ToS Terms of Service

TOU Terms of Use

TPU Tensor Processing Unit

URI Uniform Resource Identifier

UX User Experience

Abstract

Vinyl is back!

Declaration of originality

I hereby confirm that this dissertation is my own original work unless referenced clearly to the contrary, and that no portion of the work referred to in the dissertation has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Intellectual property statement

i The author of this thesis (including any appendices and/or schedules to this thesis) owns certain copyright or related rights in it (the "Copyright") and s/he has given The University of Manchester certain rights to use such Copyright, including for administrative purposes.

ii Copies of this thesis, either in full or in extracts and whether in hard or electronic copy, may be made *only* in accordance with the Copyright, Designs and Patents Act 1988 (as amended) and regulations issued under it or, where appropriate, in accordance with licensing agreements which the University has from time to time. This page must form part of any such copies made.

iii The ownership of certain Copyright, patents, designs, trademarks and other intellectual property (the "Intellectual Property") and any reproductions of copyright works in the thesis, for example graphs and tables ("Reproductions"), which may be described in this thesis, may not be owned by the author and may be owned by third parties. Such Intellectual Property and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property and/or Reproductions.

iv Further information on the conditions under which disclosure, publication and commercialisation of this thesis, the Copyright and any Intellectual Property and/or Reproductions described in it may take place is available in the University IP Policy (see `http://documents.manchester.ac.uk/DocuInfo.aspx?DocID=24420`), in any relevant Dissertation restriction declarations deposited in the University Library, and The University Library's regulations (see `http://www.library.manchester.ac.uk/about/regulations/_files/Library-regulations.pdf`).

Acknowledgements

I would like to extend my gratitude to the noble mahogany tree, whose sacrifice provided not only the material for a Welsh love spoon - by which I proposed and became engaged to my beloved fiancée - but also the offcuts that found purpose in the physical interface of this project. Your contribution to both my personal and academic life has been truly invaluable.

Also, to my close friend Joshua Bond's dissertation [1], which I have yet to finish reading - but I am sure it is great.

1 Introduction

1.1 Background and motivation

1.2 Aims and objectives

1.3 Report structure

This report consists of seven chapters:

Chapter 1 presents an introduction to the project.

Chapter 2 presents the background behind this project, ...

Chapter 3 presents details on the design ...

Chapter 4 presents details on the implementation ...

Chapter 5 presents the results ...

Chapter 6 presents an evaluation ...

Chapter 7 presents a discussion of the conclusion, limitations, and possible improvements of the project.

2 Background and Literature Review

2.1 Overview of Related Systems

Whilst the creation of a digitised turntable software is a rather novel idea, it is important to consider where this sits in the existing landscape; to understand important technologies and design decisions used in similar projects, in order to best utilise them.

2.1.1 Vinyl Systems

> *“Vinyl is back!” [2]*

In 2023, UK vinyl sales reached their highest level since 1990 [3], confirming the ongoing “vinyl revival” [4] (see Figure 1). Initially dismissed as a short-term trend when it emerged in 2008-2009, this resurgence has persisted, highlighting a renewed interest in physical music formats. Understanding the motivations behind this revival is crucial for informing design decisions, particularly from a UX perspective, as vinyl collectors constitute a key target audience.

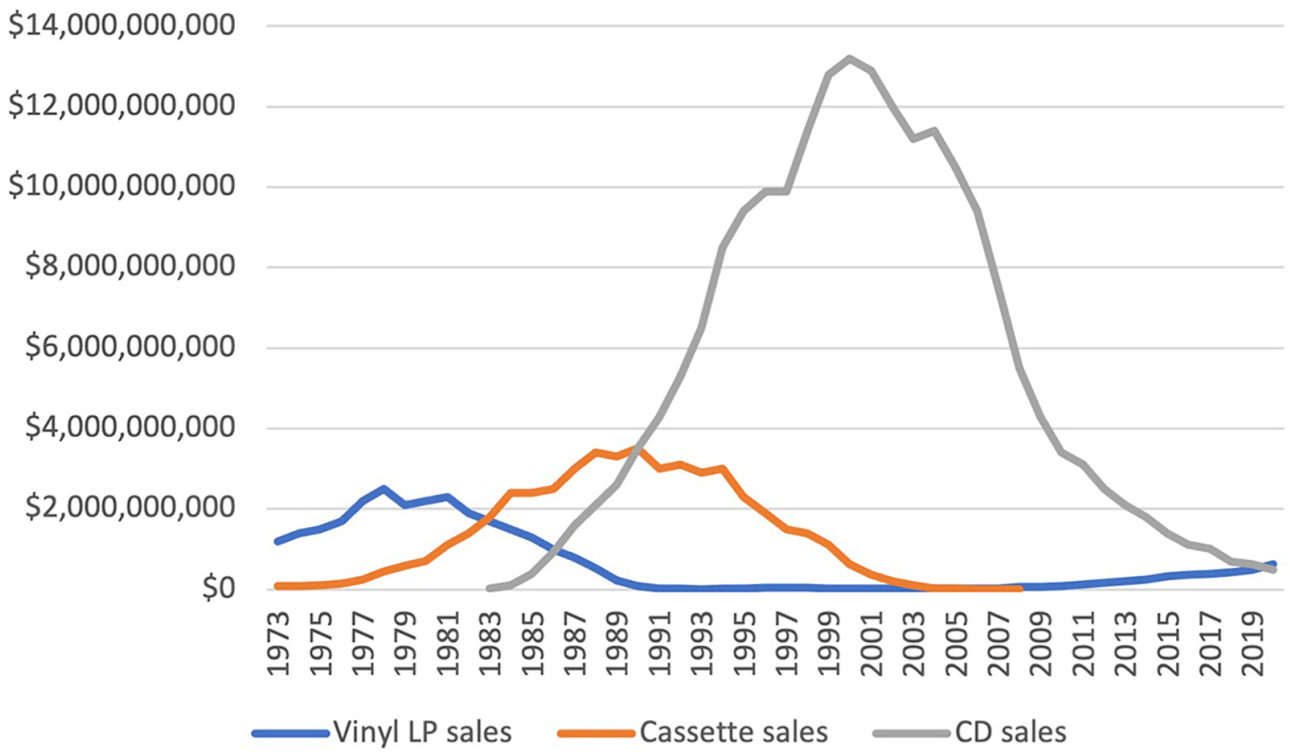

Fig. 1. Vinyl LP, Cassette, and CD Sales Revenue (1973–2020).

Sourced from Journal of Popular Music Studies [4]

Although now taken for granted, records — and their predecessors, Edison's cylinders — transformed music from an ephemeral experience into a reproducible medium. Before recording technology, music was transient and confined to its place and time of performance, unable to be stored or shared beyond a live setting. While compositions could be transcribed into musical notation, each unique performance could never be heard again once it ended, unlike with visual art, where many original works from as far back as antiquity still survive [1].

This revolution in music consumption not only shaped the modern music industry but also cemented vinyl's cultural significance. By making music ownable and replayable, it changed the way people engaged with it, fostering a more personal and enduring connection. Its enduring appeal, even in the digital age, suggests that its value extends beyond convenience, tapping into a deeper connection with music as a tangible experience.

Aesthetics and Emotional Appeal Nostalgia plays a significant role in vinyl's resurgence. Statista data suggests the revival is primarily driven by two age groups: those aged 55+ and 25–34, with other demographics showing less engagement [5]. Older generations retain direct sentimental ties to the medium, while younger consumers are drawn to its cultural legacy.

Despite the convenience of digital music, many consumers find it impersonal. In the past, music was a shared experience, often centred around a single household phonograph. Today, listeners frequently engage in isolated listening experiences [6], with it being commonplace for multiple people in the same room to listen to different tracks at the same time. To an extent, music used to demand focus. You could not skip or replay a track without having to carefully reposition the needle.

Some seek to reclaim the intentionality of music consumption, preferring a medium that encourages engagement rather than passive background listening [7].

To many, music has become hollow- especially as music streaming services have easily monopolised the digital sphere [6]. Whether people are yearning for the experiences of their past, or just a breath of 'fresh air', many are turning to vinyl to do so.

People are seeking community around their music and even its medium. Reddit's r/Vinyl, as of 2025, has over 2.2 million members. Whilst vinyl is still a relatively niche option, the social aspect of the internet means that, today, people are not confined to geographical constraints, and so, no matter how niche an interest is, they will be able to find like-minded people to connect to.

Physicality and Ownership Digital media ownership has become increasingly precarious, with consumers often purchasing revocable licenses rather than tangible assets [8]. This transition has upset many people, with there being many calls to bring back genuine ownership [9], with legislation even being passed in California to make this fact more transparent to consumers [10].

If a streaming vendor stops serving a particular piece of music, then that album can be lost to the public forever [11]. However, if a consumer actually owns the physical discs or digital audio files, then they can ensure that they can listen to their audio, regardless of whatever licensing disputes may lead to the removal of digital media in the future (see [12]).

Furthermore, there are also concerns that streaming platforms often provide artists with minimal financial compensation, leading some consumers to purchase physical media as a means of direct support [6]. This means that there is a demographic of people who both own physical vinyls, but still make use of digital streaming services. Many even purchase physical vinyls despite not owning a device to play them on [13]. In addition, the act of gaining a physical good in support of an artist can even result in a psychological feeling of proximity to their idol [6].

Additionally, vinyl's finite nature contrasts with the boundless availability of digital tracks, making collections feel more meaningful and curated.

Audiophilia and Sound Quality Another significant factor is the quality of the music being offered.

> ***audiophile****: a person who is especially interested in high-fidelity sound reproduction. [14]*

Vinyl is often perceived as superior in audio quality due to early digital compression limitations, such as the distortion issues highlighted in "Tom's Diner", wherein the a cappella's clean, isolated vocals revealed artifacts and distortions when encoded in early MP3 formats **TODO**. Modern digital formats generally surpass vinyl in fidelity, particularly in bit depth and dynamic range. However, whilst advances in manufacturing storage drives have somewhat mitigated the need drastically to compress files, there are still valid use cases where extreme compression may be needed, such as for streaming audio on a low-quality network, which may cause the audio to sound worse than on vinyl.

Additionally, as a physical format, vinyls are prone to being scratched and having physical deformities which affect the playback quality. Vinyl enthusiasts appreciate the medium's imperfections, which are thought to add warmth and character. No two discs sounds exactly the same, whereas digital copies are utterly identical. This also creates a sense of personal ownership with vinyls - not only does the consumer own the physical disc itself, but they own their precise and unique version of it.

It is important to note that many contemporary vinyl releases originate from digital masters, meaning potential losses in fidelity depend on how well an album is adapted to the format. Several inherent limitations affect vinyl playback, including: duration constraints, due to physical disk size; track sequence issues, as resolution gradually degrades towards the inner grooves; and RIAA equalisation, which alters frequency response to accommodate physical limitations of the medium [15]. Additionally, stereo information handling differs from digital formats, as vinyl relies on lateral and vertical groove modulation, which can introduce crosstalk and phase issues [15]. These factors, if not carefully managed during production, may compromise the listening experience - meaning modern tracks may often perform better in their original digital forms.

Conclusion While digital audio services offer notable convenience, the enduring vinyl revival demonstrates that many users still value tangible, nostalgic experiences. Nostalgia and charm play a crucial role in this appeal, making it essential to design with these emotional connections in mind. By combining the strengths of both physical and digital formats, a system can provide a richer, more meaningful user experience that aligns with modern consumption habits while preserving the authenticity and personal connection that vinyl enthusiasts cherish. A significant portion of consumers actively engage with both streaming services and physical media, indicating that a hybrid approach has a viable audience.

2.1.2 Image Recognition

Y. LeCun and Y. Bengio, 1995, "Convolutional Networks for Images, Speech, and Time-Series". Brain theory neural networks, vol. 3361

Image recognition is the creation of software and tools which can be used to identify objects, places, people, etc. in digital images, which has existed since at least 1946 [16]. However, in this brief time, the field has undergone several drastic changes as technology has advanced [17], and is still be redefined in the present day, particularly with the arrival of machine learning approaches, with new implementations being utilised across the field (e.g. [18], 2025).

Traditional Methods Before the advent of deep learning, image recognition primarily relied on manually crafted feature extraction techniques. Classical methods included edge detection, template matching, and statistical pattern recognition. Notable feature descriptors such as Scale-Invariant Feature Transform (SIFT), Speeded-Up Robust Features (SURF), and Histogram of Oriented Gradients (HOG) played a significant role in object detection and classification [19]. However, these approaches were often limited by their inability to generalise across variations in lighting, scale, and

occlusions. The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) served as a benchmark for evaluating the effectiveness of traditional and emerging techniques [20].

Emergence of Convolutional Neural Networks The introduction of convolutional neural networks (CNNs) marked a paradigm shift in image recognition. Early work, such as LeNet-5, demonstrated CNNs' potential [21], but it was the breakthrough of AlexNet in the 2012 ImageNet competition that solidified their dominance [17]. Subsequent architectures, including VGG, ResNet, and EfficientNet, further improved performance by introducing deeper networks, residual connections, and optimised convolutional layers [22]. These advancements enabled significant improvements in tasks such as object detection, facial recognition, and medical image analysis.

In the context of vinyl cover art recognition, CNNs offer a powerful solution for identifying album artwork despite variations in artistic style, degradation, and distortions. However, a key challenge in training CNNs for this task is the limited availability of labelled datasets. Unlike large-scale image classification datasets like ImageNet, curated datasets for album cover recognition remain relatively small. To address this limitation, data augmentation techniques—such as random cropping, rotation, colour jittering, and synthetic distortions—help improve model robustness by simulating real-world variations [23]. There has even been research done recently into using stable diffusion techniques to facilitate fully artificial data augmentation, with generative AI [24].

Multi-Headed Networks and Consensus-Based Recognition A limitation of conventional CNN models in image recognition is their reliance on a single decision pathway, which can lead to misclassifications when dealing with visually similar or degraded images. One approach to mitigate this issue is the use of multi-headed neural networks, where multiple CNN branches extract different features and contribute to a consensus decision. This technique allows the network to assess multiple aspects of the image, such as texture, dominant colours, and key object structures, before making a final classification [25].

By integrating multi-headed architectures with ensemble strategies, models can achieve higher classification accuracy and robustness, particularly when handling ambiguous or visually noisy inputs. This method has been successfully applied in fine-grained classification tasks and could be adapted for vinyl cover recognition by leveraging multiple specialised feature extractors that assess distinct album cover characteristics.

Current 'State of the Art' Visual transformers currently represent the state-of-the-art in image classification, significantly outperforming traditional CNNs on large datasets. However, their performance heavily depends on large-scale data availability; with limited or specialised datasets, such as vinyl cover recognition, transformers may perform poorly or be impractical. Thus, CNN-based architectures or hybrid approaches remain more suitable in scenarios with smaller, niche datasets.

A key challenge is ensuring models generalise effectively to unseen data, avoiding overfitting. Techniques like dropout, which randomly deactivate neurons during training, improve robustness by preventing reliance on specific features. Additionally, splitting datasets into training, validation, and

test subsets enables monitoring model generalisation, though careful consideration is required to balance partition sizes and representativeness, whilst also avoiding biases. Data augmentation (e.g., rotations, lighting adjustments) further enhances performance by artificially expanding the training data, and improving generalisation by simulating real-world variations. However, larger augmented datasets increase computational demands, making GPU or TPU optimisations essential for efficiency.

2.2 Legal and Ethical Considerations

The training and use of machine learning models for image classification requires the acquisition and processing of data, which, in order to effectively handle the cover art of existing albums, necessitates obtaining and using their copyrighted artworks. This raises both legal and ethical concerns, particularly regarding compliance with UK copyright law and exemptions. This section examines the legal basis for dataset usage, fair dealing exemptions, and the ethical implications of using copyrighted material in an academic AI project.

2.2.1 Copyright Law and Fair Dealing

Under the *Copyright, Designs and Patents Act 1988* [26], creative works, including album covers, are protected from unauthorised use, reproduction, distribution, and modification.

However, UK law also provides a key exception known as Fair Dealing, which allows limited use of copyrighted material under specific conditions, although such exemptions are only granted under very specific circumstances and for a very limited scope of use. Importantly, it is not a rigid rule but a context-dependent legal doctrine, evaluated on a case-by-case basis. The law does not explicitly define what qualifies as fair dealing; instead, courts assess whether a use is reasonable and justified based on the combination of several established legal factors.

One of the most relevant exemptions is non-commercial research, as outlined in *Section 29(1)* of the CDPA:

> *Fair dealing with a work for the purposes of research for a non-commercial purpose does not infringe any copyright in the work provided that it is accompanied by a sufficient acknowledgement. [26]*

This indicates that non-commercial academic research can be exempt from copyright infringement if proper attribution is provided. However, the applicability of this exemption depends on further and additional factors, such as the amount of material used and its impact on the copyright holder's market.

2.2.2 Use of Artworks in Model Training

Album covers are protected by copyright as highly creative works. Any reproduction or modification is typically restricted without permission from the copyright holder. As such, the bar needed for fair

dealings is very high when dealing with such artwork. However, training a classification model may qualify for fair dealing, provided certain conditions are met.

A key legal question is whether machine learning training qualifies as "*computational analysis*" under *Section 29A* of the CDPA, which states:

> *(1) The making of a copy of a work by a person who has lawful access to the work does not infringe copyright in the work provided that—*
> *(a) the copy is made in order that a person who has lawful access to the work may carry out a computational analysis of anything recorded in the work for the sole purpose of research for a non-commercial purpose, and*
> *(b) the copy is accompanied by a sufficient acknowledgement (unless this would be impossible for reasons of practicality or otherwise). [26]*

This expresses that there is a strong argument for legal use of copyrighted materials in creating a computational model which classifies data by comparisons of analytically-derived embeddings.

It is, however, important to consider whether image processing qualifies as analysis under this law or whether this interpretation is too broad, given that past applications of computational analysis have predominantly involved text, and that image processing of this kind is still relatively new. Since there is no clear legal precedent on this specific topic (with changes being in the process of being made [27]), further legal clarification over the next few years will be necessary to definitively confirm or deny its applicability to image-based AI models. But, in the time before then, it can be used as a basis.

It also reiterates the need for attribution, but, notably states that this is only required in cases where it is feasibly practical.

Given these factors, classification likely qualifies under Fair Dealing because the model does not generate new images but merely classifies existing ones. This distinction is important, especially given recent scrutiny of generative AI models like OpenAI's *DALL·E* [28], [29], which create derivative works rather than merely labelling.

None of this is clear-cut, however, as we are still in an uncertain time with the law not having been stabalised after the emergence and mass adoption [30] of these new technologies. It is worth noting, however, that there are currently proposals for UK law the explcitly allow the use of copyrighted materials in such cases [27], whereas, in the US, there is starting to be legal prescendent of cases winning on the basis [31] of AI agents using copyrighted data.

2.2.3 Legal Compliance in Dataset Sourcing

Beyond Fair Dealing considerations, data sourcing must be legally compliant. According to *Section 29A* [26], only individuals with lawful access to copyrighted works may use them for computational analysis. Therefore, it is essential to determine how these images can be legitimately acquired.

Machine learning models must fully process training images in their entirety to generate a model. This requires the whole image to be either stored persistently (on disk) or temporarily (in memory).

Even if the image is only ever stored and processed in chunks (similar to how streaming providers serve video data), the overall image is eventually processed by the model. *Section 28A* outlines more leniency for cases where only temporary copies are stored, for lawful access.

It is also important to consider if entire images are required, as opposed to only sections of them. If it would be possible to achieve the desired result using only subsets of the acquired dataset, then more data would be stored and used than is justified. The legal precedent *Ashdown v Telegraph Group Ltd (2002)*, highlights that:

> *The third most important factor is the amount and importance of the work that has been taken . . . in some circumstances the taking of an excessive amount, or the taking of even a small amount if on a regular basis, would negative fair dealing. [32]*

Thus, ensuring only necessary data is used is critical for compliance.

In addition to just the handling of the data, however, its source must also be considered. There are three methods by which the training dataset could be acquired: by fetching data from an API, by scraping the data from the web, or, by manually taking the required photos (either by just me, personally, or by crowd-sourcing the images). Realistically, the first two options are most practically feasible.

There are many vendors of the cover arts of music albums. Notably, music vendors (such as *Spotify*) and music collection and review sites (such as *Discogs*) provide the album arts in a structured format where the artworks are synchronised with the albums which they belong to. However, due to the recent boom of generative AI models - and the controversy surrounding them [33] - many vendors have explicitly prohibited the use of their data for machine learning in their Terms of Services.

> *Do not use the Spotify Platform or any Spotify Content to train a machine learning or AI model or otherwise ingest Spotify Content into a machine learning or AI model.*

[34] (III.14)

> *Do not misuse the Spotify Platform, including by i. using the Spotify Platform or any Spotify Content to train a machine learning or AI model or otherwise ingesting Spotify Content into a machine learning or AI model;*

[35] (IV.2.a.i)

> *[Discogs] strictly prohibit (1) the development of any software program, including, but not limited to, training a machine learning or artificial intelligence (AI) system using the Service content*

[36] (LICENSE AND SITE ACCESS)

However, if a site has more permissive policies, allowing the training of AI models, then, as long as the images are handled appropriately, they can be lawfully accessed and used.

2.2.4 Ethical Considerations

Even if it is legally permissible to source and use these images, it is also important to consider whether or not it is ethically responsible. These images, at the end of the day, are the highly creative works of artists, whose livelihoods come from their creations [37]. Reproducing (by downloading) and using their works therefore cannot be done without serious moral consideration.

Most significantly, it is worth noting that this AI model is not generative (which is where most of the recent controversies stem from [33]), and therefore, instead of producing its own artworks based off of the images fed to it, it simply classifies them by labelling them with its prediction of their corresponding album. Therefore, whilst the model is technical derived from the artist's works, the produced work is not in competition with the additional artists - unlike generative agents [38] - and therefore should not have a negative impact on their commercial success. If anything, it is argued that this system should benefit them, by encouraging the purchasing of physical media, and garnering instances of playing their content on a revenue-generating service.

Furthermore, as this does not share or distribute the images themselves with the users, I believe it to be even more safe, as the only artefact generated from these images are a classification system which can be used by the user, but even the numerics themselves are not made accessible to the user.

And, whilst the law allows for the exclusion for explicit attribution of all involved copyright holders, this may not be ethical. However, as this is a classification system, it arguably gives some degree of implicit accreditation to the artworks used in the training process, when the predicted label is used to redirect the user to said album.

2.2.5 Conclusion

This section examined the legal and ethical implications of using copyrighted album covers in machine learning. Based on UK Fair Dealing exemptions in the CDPA 1988, I would argue that there is a solid grounding this project likely qualifies as a legally permissible use case, provided:

- It is a non-commercial, research project.
- Data is lawfully acquired from permitting sources.
- The dataset scale and usage is minimised to strictly what is neccessary.
- None of the images are shared or modified.

From an ethical standpoint, the project is distinguishable from controversial generative AI models, as it does not replace artists' work or impact their revenue streams. Nonetheless, transparency and attribution best practices should be followed.

3 Design

What did past-Jack set out to do? Here should be the broad vision, including the change of direction taken.

Initially, the project was envisioned to be a native desktop/mobile application, with a goal of supporting a range of devices. Since, for many, music is now a personal and individual consumption, bringing the idea of the vinyl system into one's pocket seemed like the most practical and intuitive approach. Furthermore, since having a portable system would allow for an incredibly large and diverse range of albums being scanned, it was decided to use streaming services and APIs in order to maximise the breadth of the available 'library' for the application. Since this initial design focused on being device-agnostic and using live audio streaming, it was decided that a web architecture was most fitting, as frameworks already exist to serve web content on various devices, and because most audio APIs/SDKs require (or work best with) these technologies. Prior experience with Android Java development and the Spotify Web and Android SDKs, specifically, also made this an appealing decision.

However, shortly into the development period, since the background research showed that there is a strong attraction of existing vinyl users for the medium due to its physical nature: it was decided to instead create a single physical device, whilst still utilising online music streaming. This would still allow it to have the convenience of a digital player with easy library availability and playing to the strengths of virtual playback for modern tracks, whilst also encouraging the ownership of physical mediums, and leaning into the physicality in its design and interaction mechanisms. Additionally, the research showed that there is an existing demographic of consumers who purchase both physical and digital copies (or subscriptions) simultaneously - making this combinatorial design viable with a target audience of consumers. Despite being a single, localised device, in order to integrate the streaming services seamlessly, it was still required to use a webstack, and so, the design became to run both the server and client on the same device, locally. Deviating from the exact technologies used in a previous project was also seen as a good opportunity to develop new and additional skills and familiarity with other technologies.

3.1 Requirements Analysis

UML Use Case Diagram

3.2 Pillars of Design Philosophy

Some core principles were adopted in the design phase:

- **Platform Agnosticism** A core design philosophy of the project was to, where possible, make the system as robust as possible by preventing it from becoming overly reliant on any one external component. This was largely done by implementing the system through 'hexagonal

programming' (or, 'plugs and adapters') techniques, allowing components to conform to common interfaces, such that they could be easily switched, on demand. In addition to just being flexible for services, it was also a priority to make the application accessible across a multitude of devices; supporting as many OSes and architectures as possible.

- **Physicality First** The device should lean into its physical nature, and, where possible, utilise this, to best make use of the nostalgia and enjoyment people have for physical vinyl players.

- **Digital Completeness** Whilst the system was designed to function with physicality first and foremost, a secondary core design principle was to make everything the physical components could do also be achievable through digital means. This is particularly important to consider so that those with physical impairments are still accommodated for and not marginalised by the system.

- **Open Source** Whilst not inherently imperative for most design decisions, with a firm belief in the importance and value of the open source community, it was decided that this project would a contributor. Under the confirmation that publicising the code for the project in its entirety would not be in infringement of academic malpractice rules, the repository was made public. Additionally, open-source offerings were favoured, where possible, when selecting which services and technologies to use.

- **Robustness and Maintainability** Writing clean, modular, and maintainable code was emphasised from the outset. The use of established design patterns, adherence to the Single Responsibility Principle, and rigorous unit testing practices help ensure reliability, ease of maintenance, and facilitate future enhancements.

- **Ethical Responsibility** The project was developed with ethical considerations at the forefront, particularly regarding copyright compliance, transparency, responsible data usage, and awareness of broader social impacts. This also extends to having an intentional approach in making the system secure and safe.

3.3 System Architecture

*Here should be **descriptive** elements as to what was decided for the final design.*

System Architecture Diagram 2; Note from Sean: Discogs/Spotify should be together

Data Flow Diagram; Component Diagram;

Hexagonal Architecture (Ports & Plugs); (modularity = Single Responsibility Principle)

Single-page application

The final architecture design was to create a web application with a single, central server running on the physical device, that could be interfaced with from both physical analogue controls, in addition to exposed REST endpoints. This same device would also locally run a web client in order to stream the audio and display visuals for the user, communicating through the server with both traditional server calls, but, also, a bi-directional Websocket connection, allowing both the server and client to instantaneously broadcast to each other.

The device was designed to use a motor to spin a disc during playback – to replicate how an original vinyl system spins the vinyl itself – and the host client's interface would be projected, vertically-downwards onto the system, making the physic disc be textured like a vinyl, whilst also giving live information such as the current user in a holographic-like style.

It was later decided, further into development, that the network architecture should be extended to allow for 'remote clients' (that is, devices other than the main one running the server) to also connect, in order to allow easy and communal remote control of the device. In order to accommodate this, the server was updated to handle multiple websocket connections simultaneously (see Figure 2).

3.3.1 Design Choices

*Here should be the **justificatory** elements for the choices made: particularly the more broad, or weird, or niche choices. Very specific logical choices should be in their specific components.*

For example:

1. *Why use a web approach for a localised device?*
2. *Why use an unorthodox 1-1 Websocket approach for client-server calls?*

Also things such as 'point of truth handling', etc.

Is it a good idea to write this explicitly like an FAQ section, with the question stated, and then answer given?

Why use a web approach for a localised device? Despite being an application initially designed to run on a single-device, a webstack solution was still used. Whilst this seems slightly unintuitive, it maximised the options when selecting a streaming service to implement, since, as streaming services, they rely on web technologies. In addition to this, it made the application inherently device-agnostic. Furthermore, when the decision was made to allow remote device to connect, the existing web architecture made this a seamless addition, since it was already configured, and network hosting removed the need for any per-device configuration.

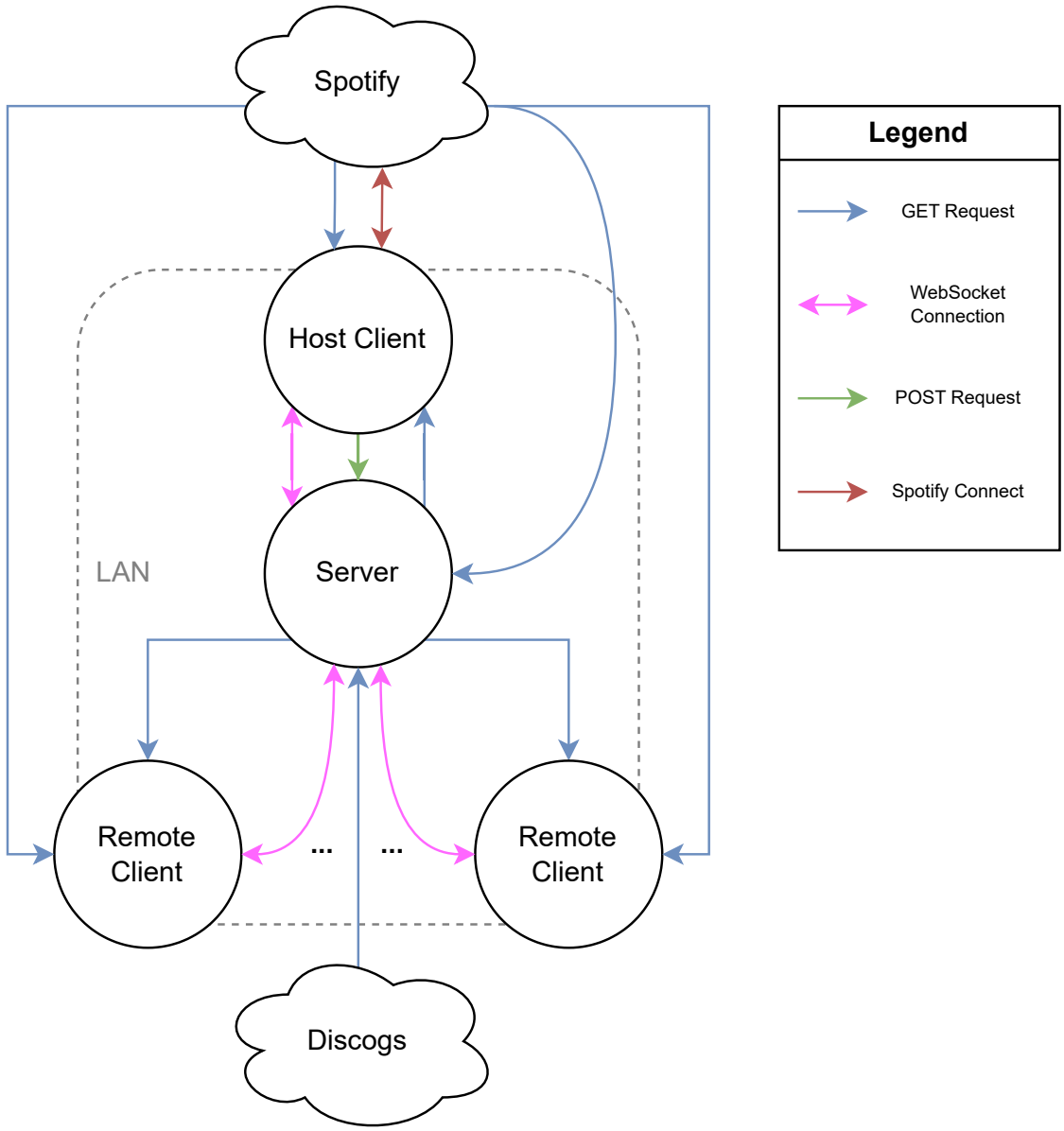

Fig. 2. Network diagram of system.

3.3.2 Technology Stack

This project utilises a range of technologies, each chosen for their relevance to the project's requirements:

Is this actually needed? All of these things are covered below. But, it seems good to have a concise high-level summary?

- **Music Vendor:** Spotify Web Playback SDK — used as the main audio stream provider. Spotify was chosen due to the existing Premium subscription requirement, unlike alternatives

(e.g. Apple Music). Prior experience with the SDK also influenced this choice.

- **Frontend:** React, TypeScript — used to support streaming SDKs. TypeScript ensures type safety, while React provides efficient reactivity to data changes from multiple sources.
- **Backend:** Python, FastAPI — selected for its strong machine learning library support and familiarity. FastAPI was chosen for its simplicity, lightweight nature, and ease of use.
- **Build Tool:** Bun & Vite — Bun was chosen for its speed, with Vite used on top to enable rapid testing and network hosting.
- **Hardware:** Raspberry Pi 5 — a Raspberry Pi board was chosen for easy GUI interfacing with simultaneous modularity for physical components; the latest board available at the time was used to maximise performance of the neural network.

3.4 Hardware

It was decided to run this system on a Raspberry Pi 5, in order to have a flexible and easy-to-use system, whilst still having decent processing power for the neural network. Microcontrollers such as Arduino boards were considered, however, since the software was designed to be usable on any reasonable device, it needed a device that could run a general-use OS and web browser. The NVIDIA® Jetson Nano™ Developer Kit was considered, as it is a powerful AI-oriented chipboard, featuring an NVIDIA Maxwell architecture with 128 NVIDIA CUDA® cores. For running a neural network solution, this architecture is significantly more powerful than the Raspberry Pi 5's equivalent VideoCore VII GPU, which is primarily for graphics and not general-purpose GPU compute like CUDA. However, this superior computing comes at a more significant price than the Pi, and, whilst it would be more capable at running ML models, using the less specialised Pi necessitated good optimisation practices, better ensuring that the final product should function well on a range of systems.

However, rather than just running the device, aesthetics were considered. In order to best utilise nostalgia, the device was designed to make use of a stylish mahogany wood with brass controls. Research shows that, in one survey, wood is the most frequently cited material in nostalgic household items, appearing in 34% of nostalgic objects (with metal second at 21%) [39]. These materials were ubiquitous in classic mid-20th-century audio equipment, so they instantly call to mind "the charm of a bygone era" [40].

3.4.1 Hardware Components

During the design phase, it was crucial to ensure the chosen hardware components could be accommodated by the Raspberry Pi's limited number of GPIO pins, some of which have specific functionality constraints (see Figure 3).

It was decided that the core functionality could be covered by a motor driver, a DC motor, a rotary encoder switch, a button and a single camera (see Table 1). This would allow the device to capture images to process through the camera, which can be manually triggered with the button. The

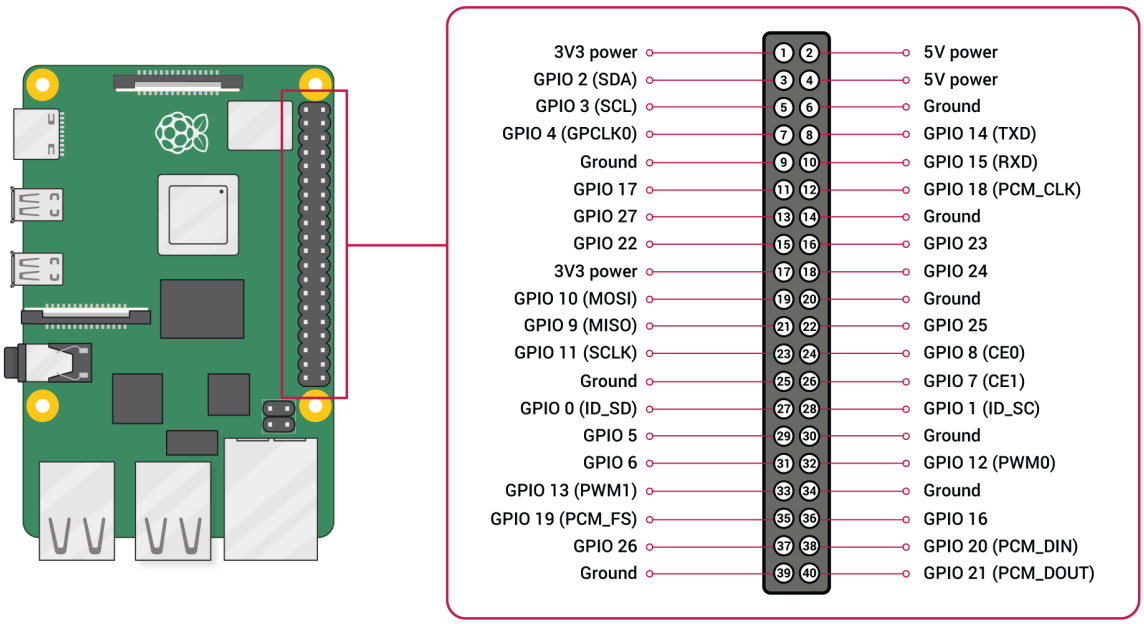

Fig. 3. Pinout configuration of Raspberry Pi 5

Source: Raspberry Pi Documentation

rotary encoder switch allows rotational vector input (direction of clockwise or anticlockwise, with magnitude), in addition to a single press-button – these elements can be combined to fully handle playback control (play, pause), track skipping (previous, next), and volume control, by using a combinatorial solution that tracks rotation in relation to if the button is pressed. For example:

- **Knob rotated, button not pressed**: Change volume up or down, depending on direction and amount.
- **Button pressed, no rotation**: Toggle pause/play.
- **Knob rotated, button pressed**: Change track to previous/next, depending on direction.

However, ideally some additional features were desired, such as having a more physical interaction mechanism, using a hinge switch; a rotary encoder to monitor the device's motor; and a secondary camera to capture images of the back of a vinyl, as well as the front (see Table 2 for details). Having a large 'arm' that hinges and requires more physical movement would allow for users to have both a more simple mechanism for controlling the pause/play state of the system, whilst still allowing the freedom to choose to use the knob, if desired.

Additionally, adding a rotary encoder to the motor would allow for stall detection, which can prevent burnout if the motor is being forcibly stalled, increasing the longevity of the component.

As per Table 3, the above core and optional requirements can all be accommodated by the board's available pins and ports, whilst leaving a reasonable number of redundant pins in case of any issues that arise, or potential expansions. The exception to this is 5 V power pins, which would be at

Table 1. GPIO Pin Requirements for Core Functionality

Component	Required Pins	Note
DC Motor (via Driver)	1 × 5 V	Motor driver power
	1 × GND	Ground
	1 × 5 V	Motor power
	2 × GPIO	Direction control
	1 × GPIO (PWM)	Speed control
Rotary Encoder Switch	1 × 3.3 V	
	2 × GPIO	Rotation data (CLK, DT)
	1 × GPIO	Button input (SW)
	1 × GND	
Button (Camera Trigger)	1 × GPIO	
	1 × GND	
Camera	1 × USB or Ribbon Port	

Table 2. GPIO Pin Requirements for Optional Components

Component	Required Pins	Note
Rotary Encoder (Motor Stall Detection)	1 × 3.3 V	Sensor power (VCC)
	2 × GPIO	Rotation data (A, B)
	1 × GND	
Hinge Switch	1 × GPIO	
	1 × GND	
Secondary Camera	1 × USB or Ribbon Port	

full capacity; though, if there is any issue or need for more of these, the motor can instead be powered through the motor driver with an external battery, freeing up a pin.

Table 3. Summary of GPIO and Port Requirements vs. Availability

Pin / Port Type	Core	Optional	Total Required	Available see Figure 3
5 V Power	2		2	2
3.3 V Power	1		1	2
GND	3	2	5	8
GPIO (Standard)	6	3	9	28
GPIO (PWM-capable)	1		1	4
USB or Ribbon Port	1	1	2	2+*

*Raspberry Pi 5 supports 4× USB + 2× camera ribbon ports.

3.4.2 Operating System

Raspberry Pi OS (or just RPiOS; formerly Raspbian), is an operating system produced by the the Raspberry Pi Foundation as the official OS for their chipboards. This makes interfacing with components such as GPIO pins seamless and easy. However, in order to make the system as hardware-agnostic as possible, a standard Ubuntu kernel was used for the project. This forced solutions to not rely on native Raspberry Pi interfaces, and so, whilst it made certain aspects more challenging, it means than the project is not locked to the RPi hardware.

Additionally, Ubuntu is fully open-source, whereas RPiOS has some proprietary aspects.

For running the interface, Firefox (an open-source non-chromium browser), as Ubuntu's default browser, was used for primary development. However, the application was also periodically tested on chromium, to ensure compatibility with other browsers.

3.5 Front-end

3.5.1 Primary User Interface

- Physical user interaction controls

UI Wireframes / Mockups

The initial aim to create a device-agnostic platform made web technologies an ideal choice, and out of the available frameworks, React was selected due to having good support for reactive UI updates from multiple sources. This feature is especially useful, as the host client application is responsible for interfacing with the streaming service(s), and needs to handle state changes from not only its own UI controls, but also from the vendor itself, and from the server's websocket connection which relays commands from the physical controls and from remote devices (see Figure 4 for the different sources of state changes). In addition to this, prior experience with the framework was also an influence.

3.5.2 Audio Playback

In order to keep with the goal of being platform-agnostic, all API connections for the music playback and metadata retrieval was done through object-oriented programming interfaces and abstract classes, forcing all implementations to inherit and conform to these, enforcing uniformity and allowing the interfacing system, such as the host client, to seamlessly use any of them, as a black-box system.

However, for this project, only one external streaming API was selected to be implemented. After considering various options, Spotify was chosen due to past experiences with the system making it faster to integrate. In addition, most services required a subscription with the service in order to stream them; a pre-requisite already met for Spotify. For these reasons, Spotify was chosen for the audio streaming in the front end component, and thus, the corresponding back-end systems also used Spotify.

3.5.3 Minimal UI

Once it was decided to create a single physical device, the emphasis became to use physical controls over virtual UI buttons, where possible. A primary goal was to make the interface be entirely usable without the need for a mouse or keyboard (or even a screen), once configured. In order to

emphasise this, the host client system was designed to be as minimal in UI components as possible, such that it would essentially be in a read-only state – only showing data, and not controls, unless explicitly requested. To achieve this, buttons and sliders are hidden on the host client, unless the mouse was recently in motion.

However, once the user indicates they are interfacing through the client, by moving the mouse, it was important to ensure that the UI offered no less control and/or information as the physical controls did.

In addition to playback controls, the host client also has access to settings, which are fed to the server. These settings control who can remotely access the system (if at all), as well as making the motor component optional by disabling it. These options are provided exclusively on the host system.

3.5.4 Remote Clients

- for convineince, the need for (33-45) adapter can be cited

During development, it was decided to not only allow a web browser to locally connect to the server, on the host device, but to expand the system to allow 'remote devices' such as external mobile phones and computers. This was decided on the basis of both accessibility and general convenience. As people with physical conditions could struggle to use physical controls, a remote connection from a phone would allow them still be able to operate the device. In addition, this also provides technological accessibility, as those without the required hardware (such as a camera) would be able to now use their phone's camera, to scan albums, as needed.

This could also be used to bring a further communal aspect, as multiple people could connect to, and use, the device simultaneously, allowing an authorised individual to control playback or enqueue a song to share with the host and others. An additional benefit being that a user could upload a photo taken when away from the host device, allowing the initial portability design to not be entirely lost, even with the now-static system.

Rather than implement a separate front-end or routing system, the client is a single-page application, using calls to the server to determine if the device is the host or external. The client then displays the correct form of UI to the user.

The remote client UI includes buttons to play, pause, skip and return to previous tracks; it provides a volume slider; displays the album art of the track; allows scanning of an album; and even displays to the user the host's library of previously-scanned albums, which can be set to play individually, or shuffled as a whole collection. These controls allow the remote user to do everything a user physically interacting could.

3.6 Back-end

In order to allow multiple devices to connect with the host device, a centralised server system was needed. Not only would this server be responsible for handling communication between its clients, but it would also need to handle the physical control system, the album detection logic (discussed separately in Section 3.7), authentication (see 3.8) and any additional logic.

Python was selected as a robust and easy-to-use high-level language to best accommodate all of these; especially since it has a very rich suite of machine learning libraries. FastAPI was selected as the framework for the server, using Uvicorn, as this is known for being fast and lightweight, ideal for the SBC.

The server was created using the singleton pattern, to ensure that the machine learning model(s) are only initialised a single time at startup and across all requests, for optimal performance. Whilst parallelisation and scalability were not goals for this project, in order to achieve the goal of creating a robust solution, this bedrock layer of good practice was implemented.

The server follows the Single Responsibility Principle, meaning functionalities are broken down into classes with each one handling one, and only one, functionality. This makes the REST endpoint routing, for example, easily separable from the APIs' logic, and from the authentication protocols. This design ensures the code is more readable and maintainable.

As the server also handles the physical controls (in addition to needing to relay commands from the remote clients), the server needed to be able to instantaneously communicate with the host client. Rather than relying on inefficient polling, or the one-way, text-only approach of using Server-Sent Events (SSEs): websockets provide great flexibility in allowing the client and host to both communicate with each other, bi-directionally, with minimal delay and resource usage. This is ideal, because whilst the server will sometimes need to update the client (when a user uses a physical control, for example) there will also be times where the client systems need to update the server (for example, if the music is paused directly through Spotify, from a device not even connected to our server, the server needs to be informed of this by the host client). Using this bi-directional channel ensures that all similar communications go through the same context, and avoids needing to implement similar code multiple times, in different ways (such as if this was done with standard REST calls and SSEs).

3.6.1 Metadata Retrieval

Traditionally, when a track or album is playing, online streaming services show the cover art to the user. Since this project relies on scanning a cover art in order to play the music, it seemed redundant to broadcast the current track by showing the user the same artwork they have in their hands. So, in order to lean more into the physical vinyl concept, the server was designed to fetch the centre label of the corresponding vinyl, in order to show this on the disc, instead.

In order to fetch this, Discogs' API was used, as they maintain a carefully curated library of many images for vinyls, including high-quality images of these labels. Whilst Discogs do not allow their

images to be used in AI systems, these images are never fed into the machine learning model, and are strictly processed by traditional CV methods.

The system was designed to simply, upon playback of a track, fetch images from the Discogs library, and find the best-matching circular image, using Hough Circle detection, and then cropping the image to serve to the front-end.

3.7 Machine Learning Model Design

In light of the background research, it was decided that using a machine-learning neural network approach would be better for this project, rather than traditional tailored OCR solutions (as some albums feature no text) or feature extraction techniques (as albums vary greatly, and manual defining features would likely yield worse results). Whilst visual transformers are often the best performing models for large datasets, it was decided to use a smaller CNN, to better converge on the smaller and more specific dataset of albums.

3.7.1 Dataset Collection

In order to train the model, a dataset of albums was required. Manual collection of images would be a slow process, and could introduce bias, when testing. Whilst there is wide abundance of these images on platforms such as Spotify and Discogs, their terms strictly prohibit the use of their served content from being fed into AI systems.

One option was Last.fm, whose terms had no explicit prohibition of these use cases, and so was subject only to copyright law and fair dealings.

However, the Internet Archive's (in collaboration with Musicbrainz) Cover Art Archive provides its images, with no restrictions on AI systems, either. In addition to being a viable option, this library also provides images of the backs of albums, which none of the other surveyed systems did, and, is an open-source solution, and so was selected to be the primary source of dataset collection.

However, manual images were still collected to add to this dataset. This was essential to ensure that model validation could be done without bias (see D.4).

3.7.2 Solution Design

The high-level solution to the album detection problem was to use the model trained on album images to classify which album was being scanned. In addition, the model should ideally be able to detect when the input image is not of an album at all, for example if the camera was triggered too early, before the vinyl was placed. This could be done by feeding random images in during training as a 'null' class.

If classification for this null class worked well, the camera could essentially poll images, and only react to high-confidence changes in its state, minimising the need for a user to press a button or other control to trigger the scan.

However, rather than rely solely on the CNN mode, as a fallback system, it was decided to use the model to classify, and in cases where the model could not confidently give a prediction, it would use an OCR system, as a second attempt. In addition to OCR, barcode scanning was also decided to be used. This combinatorial approach was envisioned to make the model more robust at recognising images outside of the scope of its training data.

Decision Flowchart for Image Recognition

3.7.3 Model Architecture

Based off of the research, it was decided to use a convolutional neural network, due to their excellence in image classification tasks.

An educational approach was taken in designing the models, wherein the goal was to create a fully self-made solution with a simple architecture. Then, the model would be expanded upon and deepened, as needed, to get the best results, through experimentation informed by the literature.

The goal was to create a few different models, with differences in architecture to experiment to find results. Other than broad decisions, their architectures were not developed until during implementation. The details of each of these models is described in Section 4.4.

However, the conceptual models are described below. In order to distinguish between these models beyond just their most basic architecture, each was given a unique name. The inspiration for these names were taken from Greek mythology, and therefore each model is named after a mythological snake-creature (see Appendix G.1, for more details).

Ouroboros A standard feed-forward CNN.

At this most basic level, the goal is to simply design a model that can accurately learn information from both a small and large dataset of albums, to ensure that it can handle varying collection sizes of different users.

Since the task of learning albums from their artworks is a closed-world scenario – that is, the model is expected to classify inputs that are nearly identical to the training examples, with minimal variation, since two versions of the same album usually have identical artworks – the need for generalisation is low and the model can rely more on memorisation. This ‘memorisation-friendly’ nature of the task was the inspiration for the name Ouroboros, the self-eating serpent.

Amphisbaena A two-headed neural network, trained to capture and classify an image’s most likely artist, in addition to the album itself.

If the Ouroboros model is viable, then it was desired to experiment with adding multiple heads, so that the model could predict an artist, even if it had not been trained on a specific album. If an artist can be predicted with high confidence, but no album is overly confident, then this could be used to inform the search alongside the OCR/barcode data, etc.

Additionally, through further experimentation, this could be expanded to learn genre, or even the decade of release.

Hydra An adaptive multi-headed model, allowing the creation of new heads through knowledge distillation and consensus to learn new, previously out-of-scope classes, whilst still maintaining precision on previously trained data.

A significant issue with using a closed classification system (that is, it only trains once), is that any classes not included inside the training dataset can never be predicted. Whilst it is common practice to re-train (or fine-tune) a model, to adjust weightings based off of new data, the task of adding a whole new class to the model's options when deciding is a much more complex task.

The most simple option is to simply re-train the whole model, with the additional classes and data. However, in order to best maintain fair dealings, all images used in training were deleted after use, by the system. Therefore, unless the user was to re-download the entire dataset each time, this was not a very viable option. Additionally, it could be the case that the online source of images may have had images removed, and therefore, re-training could actually offer worse performance in some cases.

An aspiration target of model development for this project was to create a progressive neural network (PNN) that could adapt to new classes. The simple goal would be for the model to support a dynamic number of heads, and when new classes are added to train a new head using knowledge-distillation from previous heads (to prevent 'forgetting' previously trained classes), whilst still associating with the new classes. Once the new head is trained, it can be appended to the model, and a weighted consensus of the various heads would be used to determine the class.

3.8 Security Considerations

Security Architecture Diagram

Data Flow Diagram

Whilst this system is designed as a local device, with the only connectivity to external sources being trusted APIs, security was still an important consideration, and taken seriously within each phase of the project.

Handling of secure data Since this project integrates with external systems, proper authentication token handling was required. Especially when serving multiple users simultaneously. The system follows a standard RESTful token flow, wherein a client is redirected to the server through

the external service's callback, with a token that can be used to request a formal authentication token. This token is stored only in the server's memory, and is associated with the client's session ID cookie. Therefore, a user can only fetch an authentication token through the GET endpoint if they are the same device that made the request. This ensures secure data was only accessible to authorised parties.

Persistence task off-handing Some tasks require non-volatile storage, for example: storing a list of the albums a user has scanned into their 'collection'. The decision was made that, where possible, non-sensitive data should be off-loaded onto the API service being used. That is, rather than store the list of songs locally, a Spotify playlist can be created and updated as the system's 'file'. This meant all GDPR compliance needed to be met by the external vendors, rather than this system.

Local network hosting To handle remote clients, it was decided to allow the front-end runner (Vite) to use a feature called network-hosting. This feature allows LAN devices to connect to the site. Whilst opening access to certain ports to same-network devices can have consequences, as a static device, this system is designed to only be used in a secure network (such as the user's house), and so no major actions were taken, beyond ensuring firewall services did cover the relevant ports.

Command permissions It was decided that even if user A was signed in on the host device, user B should be allowed to pause, play, scan, etc. from their external device if, and only if, user A authorises this. To achieve this, not only are the required UI buttons disabled when the host has disabled remote access, but all server-side logic enforces this. The host can either disable all remote connection, or allow only the host's account to do this, or allow all devices to access the server. The validation for is a user is the host or not, is done through the session ID cookies and verifying with the external API that the associated token is for the authorised user. Therefore, this cannot be ignored by a malicious user.

API compliances An intentional effort was made to ensure that all data processing, requests, and request-frequencies were compliant with the respective APIs' terms of services, rules, and limitations.

3.9 Testing Methodology

Design of tests and evaluations; plan for unit testing, model evaluation, etc.

Does the system satisfy that physical and aesthetic desires of that the vinyl trend appeals to, whilst still offering functional convenience over original systems?

3.9.1 Validation of Effectiveness

In order to evaluate if the developed solution is effective in reaching its goals, three primary metrics will be used:

Model Performance In order to assess if the model performs adequately, formal model evaluation will be performed. Two units will be under test: the recognition model and the metadata retrieval (the centre label, etc.). In order to asses the model's performance, a fresh test set of data will be used, of front-only photos of physical covers, from which an F1 metric can be calculated. Aditionally, the centre label accuracy can be objectively assessed by running it over a known set of albums, and manually comparing the accuracy compared to that of the owned vinyls.

Usability An import goal is whether or not the system and it's physical and digital controls are accessible and useful for users. To assess this, a user-centric approach will be taken, with evaluation of survey results of people who have used the finished system.

Code Robustness In order to ensure code is functional, and to try to maximise its robustness, a full unit test suite will be developed in parallel. This suite will aim to have high code coverage.

3.9.2 Validation of Affectiveness

In addition to the practical side, the emotional value of the system also needs assessment. In order to ensure that the system has correctly appealed to its intended audience and to evaluate how well the system captures the appeal of traditional vinyl systems along with the convenience of a digital player, the user survey will also include questions regarding these aspects.

4 Implementation

During the development of the system, many details were realised in practice and some decisions and deviations were made from the design, both to improve the system and also, sometimes, in response to difficulties or challenges that occured.

4.1 Front-end

Personal ownership (predicatbale nmoise overlays), and why this wasn't done

Local file storage

The implementation of the front-end system was relatively smooth, largely due to the minimal-UI aspect of the design. The interface was designed to be practical first, with little-to-no creativity or flair being added until late in development.

Was the product was functional, some polish aspects were added. For example, when an image is taken/uploaded to the system, it is briefly displayed on the screen before being 'lifted up' to unveil the vinyl and its centre label underneath.

4.1.1 Challenges Encountered

2. Responsive / wholeness tradeoff (do we wait for the centre label before playing the audio?) 3. Media codec, DRM, etc. issues specific to Ubuntu/aarch64 OS.

Responsiveness vs. Wholeness During development one challenge that was encountered was the latency that would sometimes occur in fetching a track's metadata after the track had started playing. In order to give the user as seamless of an auditory experience as possible, it was best to let tracks play instantly, with no delay. However, this meant that sometimes the centre label would be blank briefly before 'popping in' afterwards. Generally, UI principles dictate that an aspect should be loaded in completeness before rendering, to avoid unstable changes being displayed to the user. Since the system is primarily an audio system, it was decided to favour responsiveness, allowing the system to asynchronously update the interface if and when it got the metadata. However, in order to mitigate the pop-in, once a centre label is found the image was cached in the file-store, to prevent this search process being conducted each time. Furthermore, the aforementioned image display covering the vinyl, gives more of a chance for the pop-in to be hidden from the user, for those few seconds.

DRM Issues One notable issue came from the choice of the system's hardware. Raspberry Pi SBCs use ARM-based processors. Whilst Ubuntu is designed for this, a notable absence is that Google's WideVine DRM (which is used by both Firefox and chromium-browsers alike) does not have an official AArch64 build, which is needed for a RPi. The absence of a DRM meant that Spotify's audio could not be streamed in the browser. Whilst no AArch64 build exists in the official repos, there is, however, an official build used for ChromeOS systems. It was possible to install and use this version, which enabled DRM content on the Pi. However, if this had not been available, then this would have been a serious hurdle for the project's chosen system architecture.

Additionally, there were some issues with missing media codec's in Ubuntu's default packages. This was easy to resolve, however, added more steps to the setup process, which required a greater need for documentation. This would have been minimised if using more established solutions, such the official Raspbian image on the Pi, or a proprietary OS such as Windows on an x64 board.

4.2 Back-end

The backend was developed with Python's FastAPI and asyncio libraries to maximise the performance of parallel tasks. In addition to this, PyTorch (see 4.4) and OpenCV were used to make the machine learning and image extraction processes simpler to perform.

State Management One interesting feature of the system, was that a state-propagation system essentially had to be created. Since the system can be split into three components (host client, remote client, and server), which also integrate with external systems (such as Spotify), all of these components had their own internal state which informed the system how to act, but needed to be writeable from the other components. The solution to this was to essentially create a reactive system, inspired by React's state and hooks system. All active components are linked in a linear chain (see Figure 4), therefore, other components states can be seen as dependencies. Each component is only concerned with updating adjacent components, when its internal state changes, and so, whenever a state change occurs in a component, it broadcasts this to its connected neighbour(s). This creates a real-time feedback loop, allowing changes, even from an external device using the official Spotify app, for example, to be synchronised with the motor's state. For each aspect, a single 'point of truth' was determined, and in cases where new connection are established, this was used. For example, the audio is playing through the host client app, and so, if there is a conflict of states, the host client's state is favoured.

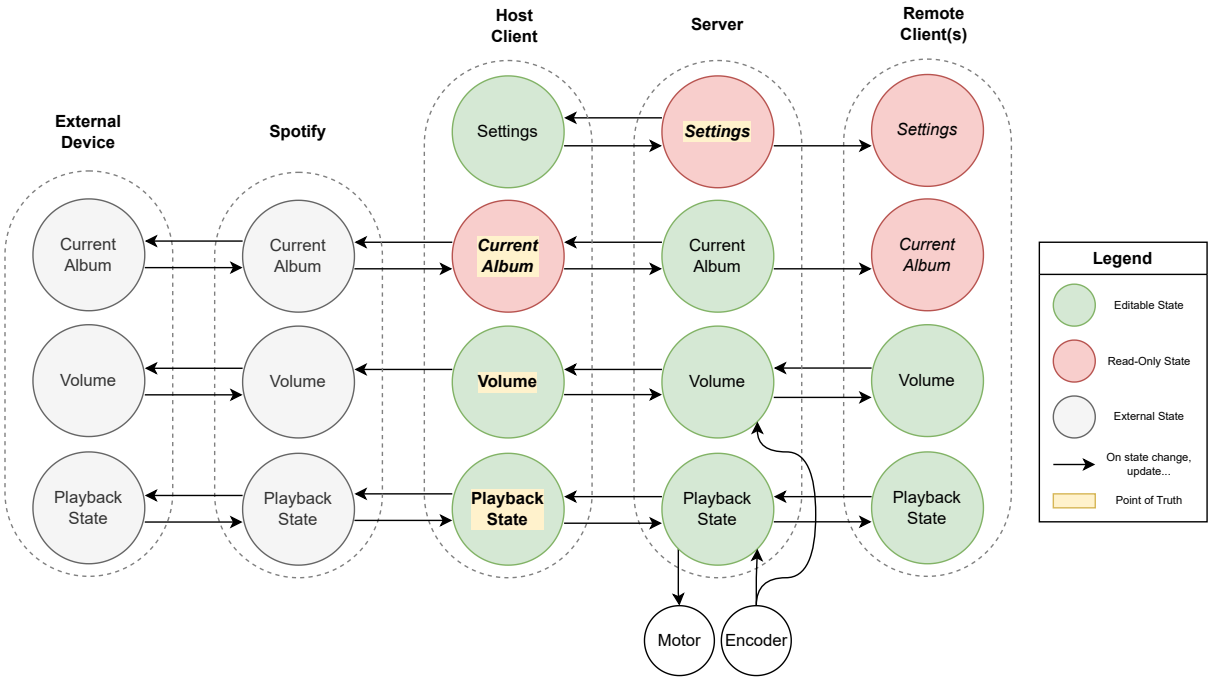

Fig. 4. Dependency diagram showing state propagation model between both system and external components, allowing real-time reactivity to state changes from multiple sources.

One complication of this feature, was that there were often superfluous broadcasts. For example, if component A updated component B, then it is wasteful for component B to broadcast that same change back to component A. However, most of these broadcasts were kept, as it essentially created a 'handshake' system where components were not only updating each other, but received

confirmation that those updates had been performed. The exception to this was in the front-end clients, where external and internal changes were flagged, to ensure that only internal changes were broadcasted. This was important because React's state system already handles state tracking, and could cause infinite feedback loops, throttling performance and the system's communication bandwidth. Additionally, since the host client interfaced directly with external components, it was best to avoid 'spamming' these components, which could result in timeouts.

Alternate Processes Another important aspect was the handling of different triggers for the same processes. For example, how both pressing the physical camera button and uploading an image from a remote device should trigger album detection and playback (see Figure 5).

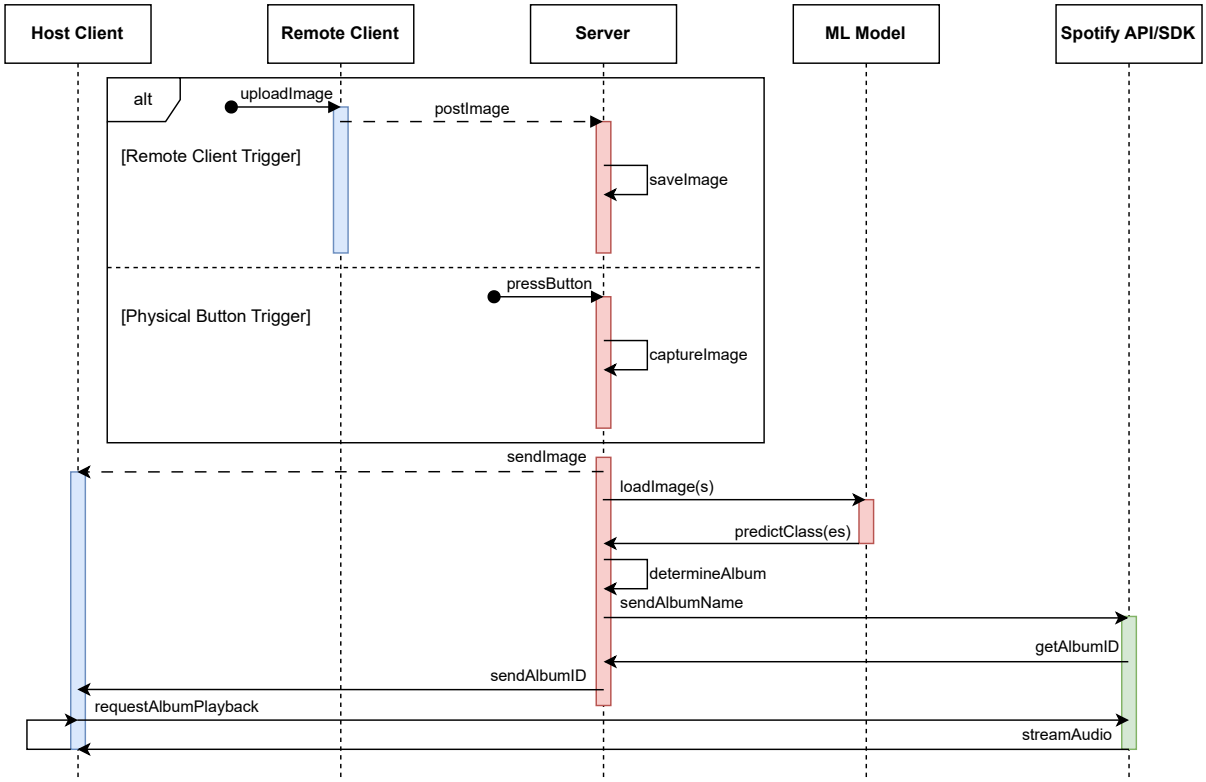

Fig. 5. Sequence diagram showing two alternative triggers (client upload and server capture) leading to image processing and metadata retrieval before sending playback instructions to the host.

In order to maximise the code's maintainability, re-using the code for the shared part of the process was of utmost importance, however since one trigger was an asynchronous REST endpoint, and the other a low-level polling of hardware. This would have been very difficult to manage if these functions were not in isolated components. Keeping with SRP practices, as decided early on, prior to any development, helped accommodate these mid-implementation changes to the system, with minimal complications.

4.2.1 Challenges Encountered

1. Change of auth flow, due to 'remote client' introduction 2. Removal of barcode detection form spec, due to camera limitations 3. [41]

Authentication handling Once the system was changed to allow multiple devices to connect, the whole authentication flow for the server had to be re-written, as it was initially built on the assumption that it would only ever be a single local connection being used. Whilst this required some work, it was a good reminder during the process to challenge the assumptions made during development, and no other changes as large as this had to be made later on in the process, due, at least partially, to this learning experience.

Removal of Barcode Scanning It was realised early on that, whilst covers do often have barcodes on their backs, which can uniquely identify the exact album: these barcodes are too low quality to ascertain information from in a image of a full album, without very high quality cameras, which this project did not want to utilise, to avoid reducing the technical accessibility of the system. It could have been setup so that, if a user's scan fails, they can optional re-scan, but this time closer to the image, so that it is just the barcode being captured, but even then, testing found that barcodes are very weak against noise and lighting conditions for general optical cameras (largely due to the fact that they only have one level of redundancy). Some vinyls have QR codes, which are much better suited for general scanning, however, there is no widely adopted system for how a record's QR code correlates to the album (oftentimes it is just a redirect to merch store, etc.). Furthermore, having to position the cover in a specific place, made the system more fiddly than convenient, and so this option was dropped entirely.

Change of Spotify Security Requirements In mid-February, towards the end of the development period, when the code was mostly frozen, other than bug-fixes, Spotify announced that it would soon be removing support for HTTP callback URIs, and only allow HTTPS connection, starting on the 9th of April [41]. This was problematic for my system, as only utilised HTTP protocols, on the assumption that it was only ever used as a local device (essentially in development mode), rather than as a traditional web-server. As previously mentioned, compliance with API requirements was an important mandate for the project, and this extended to maintaining compliance, too. However, the conversion to HTTPS introduced a fair number of regressions and issues, which took some time to correct, rather late into this phase.

It was an important reminder, that if this sudden change had not been one that could be complied with, then, the agnostic approach of the system would hopefully mitigate the difficulty in switching to another system – although this would still have been likely to large of a task for this late into the process.

4.3 Hardware

To implement the design, the following hardware components were selected:

- **Motor Driver**: TODO
- **Micro Metal Geared motor with Encoder**: 6V 75RPM 210:1

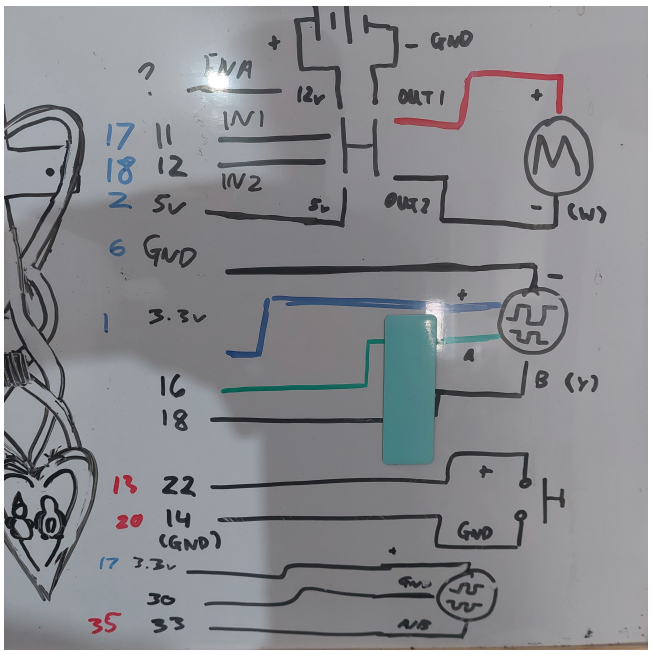

Fig. 6. Circuit diagram of the connected components (TEMP! TODO: FORMALSIE)

- **Trust Trino HD webcam**
- **KY-040 360 Degree Rotary Encoder Module**: 5 V
- **Standard button**

Additionally, the 'hinge switch' arm was created by simply wiring the Pi to isolated sides of a standard metal hinge, connected to an off-cut piece of wood, shaped similar to an arm.

Figure 6 can be referred to for the configuration of these components with the system.

4.3.1 Challenges Encountered

During the implementation phase of the project, a few problems were encountered when handling the hardware components.

'Measure Twice, Cut Once' Often, when developing software solution, it is common to use a 'trial and error' approach to make a system work. Whilst development is not a mindless task, developers will often just tweak small changes and run the system until it performs correctly, rather than fully designing every single aspect on paper first. For software, in development environments, this is safe to do and even works well as a time-efficient practice. Many technology stacks even accommodate this, with features such as hot-reloading.

However, when dealing with hardware, this is not the case. During the development of this product there were a few cases where irreparable damage was done to some components due to the use of the above software debugging technique. Whilst configuring the motor to support PWM, the

motor driver's jumper connection (which connected the PWM EN input pin to the 5V pin for constant pulse) had to be removed, to allow the PWM GPIO pin to connect. However, after removing the jumper both pins were exposed, and rather than check which one was the correct pin, a rather careless attitude was taken, just guessing which pin it was. This resulted connecting the 3V PWM GPIO pin to a live 5V current, which fried the GPIO output.

Thankfully, the damaged pin was one of two which supports PWM on the RPi 5 model, and so, the backup pin was used.

However, this led to much more caution being used when handling the hardware components afterwards.

Voltage Consumption During early development with the Pi, an unofficial charging cable was used as a power source whilst waiting for the official one to arrive. This cable did not have sufficient wattage for the Pi's specs, and, as a result, it was unable to power the device at high-load without causing the device to turn off.

To prevent this from being a total blocker, the device was configured to under-clock CPU usage, and some other things, in order to minimise the voltage draw of the device, to minimise how often this would occur.

Overheating Sometimes, during long periods of development, the device would reach very high temperatures. Due to the RPi 5's increased specs over previous models, this is fairly common when running it even for small tasks with only active cooling. This began to throttle performance during long periods of development and/or testing, however, the solution was simply to setup an active cooling solution. Thankfully, the pins for controlling fan PWM based off of CPU temperatures are separate from the standard GPIO pins, and so did not affect any of the other components.

4.4 Machine Learning Model

For implementation of the ML models, PyTorch was chosen over other libraries (such as TensorFlow, Keras, etc.) due to its superior debugging capabilities and flexibility. Given that this project focused on experimentation rather than production-grade deployment, PyTorch's dynamic computation graph and intuitive design made it a more suitable choice. Experimentation was done through Jupyter notebooks, which aided development due to easily allowing specific code snippets to re-run, and rendering results in-line.

The initial goal was to create a full pipeline that could reasonably run on the Raspberry Pi model. Although, it was known that if the dataset grew beyond a toy sample size, whilst it would be possible to train using the Pi, it would not be efficient. With this in mind, all model training and experimentation was done on desktop computer, utilising an NVIDIA RTX 3060 Ti GPU. Eventually, the idea of running the model on the Pi was dropped entirely, however, it is still technically possible, it is just not reasonable for anything other than very small models (TODO- compare results and training times of RTX vsa Pi).

4.4.1 Ouroboros Model

In keeping with the experimental learning-through-development approach, the most basic possible CNN model was initially created, as shown in Figure 7. The architecture for this was largely informed by TODO, and the first implementation was referred to as SimpleCNN.

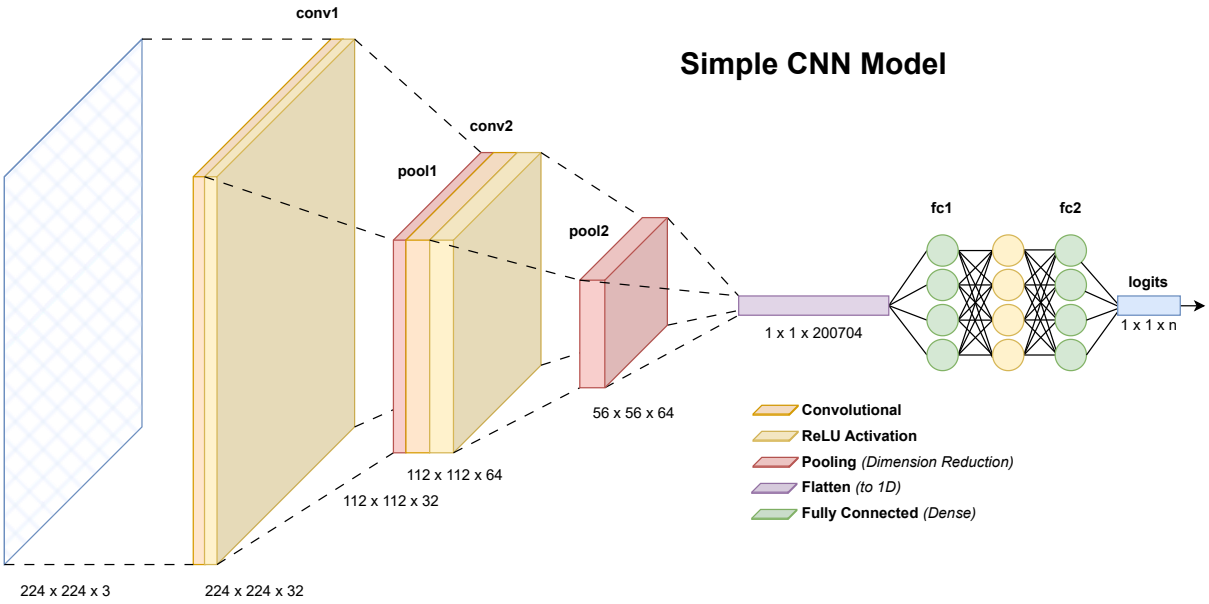

Fig. 7. Architecture for simple CNN model

The first iteration of this model (SimpleCNN 'Mini') was trained on only the exceptionally small TODO dataset, as a proof of concept. Figure 9 shows how the models accuracy increased as training progressed. Whilst the model rather quickly converged and showed good results on the training data (eventually overfitting), the validation dataset's accuracy did climb in slow intervals, but never exceeded 60%. Whilst this was a good start, it was thought that the problem with this poor generalisation likely came from the sheer small scale of the dataset.

All input data was normalised according to the some dataset statistic:

$$x' = \frac{x - \mu}{\sigma}$$

Initially, this was calculated as a channel-wide measure of the full combined dataset. However, the standardised ImageNet distribution was also used, with $\mu = [0.485, 0.456, 0.406]$ and $\sigma = [0.229, 0.224, 0.225]$. Figure 8 shows an example batch of images, normalised to the ImageNet standard.

Conforming the data to an external distribution can be a good practice for consistency and compatibility, particularly if there was a chance of later using transfer learning or pre-trained models.

The same model architecture was then used to train on a much a larger dataset (TODO). However, this model performed horrendously with the sudden increase of data. As shown in Figure 10 whilst the loss (distance of predictions from the true results) decreases rather smoothly on the smaller dataset, when trying to learn from TODO, the model very sharply overfits, and actually

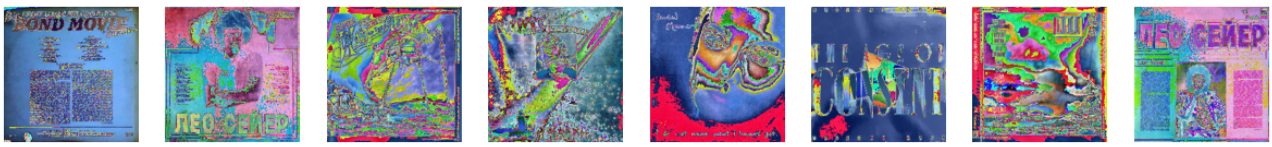

Fig. 8. Example of normalised dataset batch.

Original artworks are © their respective copyright owners These images have been processed for research/educational purposes and do not intend to infringe upon the original copyrights.

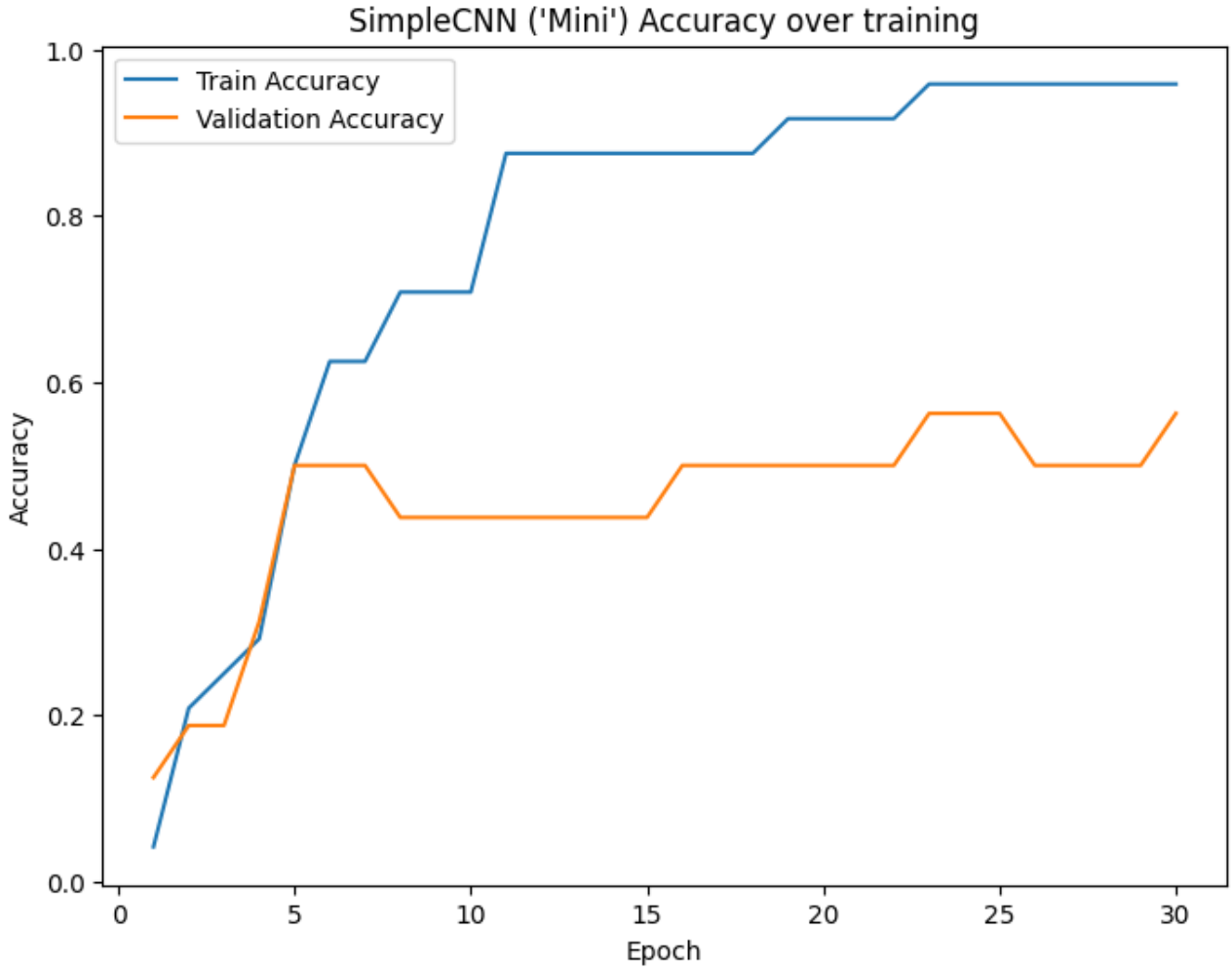

Fig. 9. TODO

Higher is better. Training of a SimpleCNN model on D.1; validated against D.4.

worsened in validation performance, never even improving beyond the smaller dataset's initial performance. Figure 11 shows the differences of the best-observed performances of this model architecture across these two training scenarios, showing that performance decreased as the dataset increased.

This model performed reasonably on small datasets, but could not scale, and so was labelled as Baby Ouroboros.

In order to improve the results for the large dataset, a deeper model would be required. Various experiments were done with various configurations and depths of model (broadly named DeepCNN),

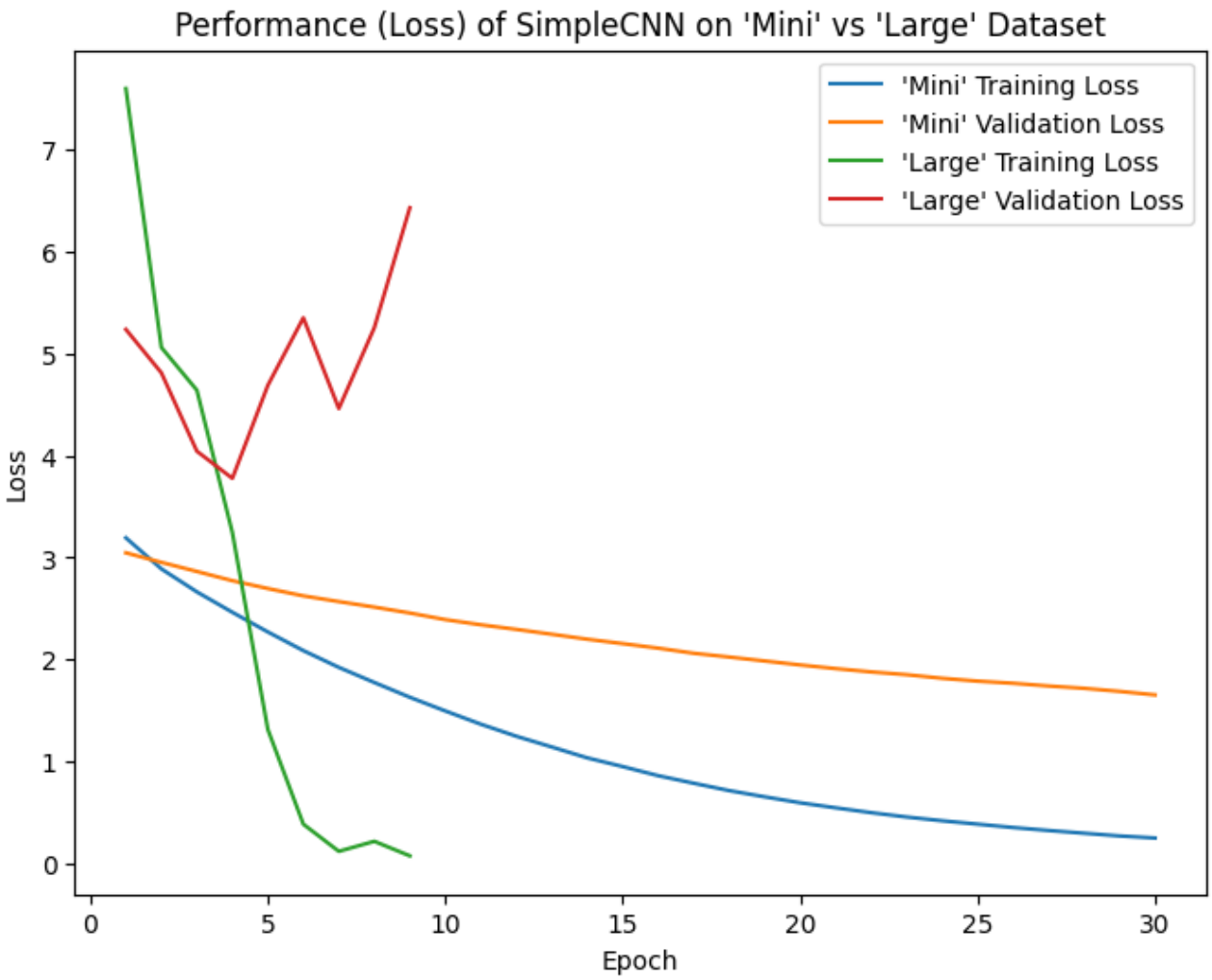

Fig. 10. Comparison of SimpleCNN architecture performance on different sized datasets

Lower is better. Trained on 'Mini' (D.1) and 'Large' (D.2) datasets; both validated against D.4.

however, as the range of albums had increased, even with a greater depth, the feature extraction mechanisms seemed to be lacking. As networks become deeper they require exponentially more data in order to learn sufficiently.

After the results of the deeper versions of Baby Ouroboros did not prove very fruitful, a new architecture was planned. Rather than trying to train a model from scratch, it was decided to trial using a feature-extractor from an existing model, which had been trained on a much larger and more generalised dataset. In order to do this, the ResNet18 model selected, which has been trained on the immense ImageNet library. In theory, a custom solution could have still been trained using this publicly available dataset, with the published parameters and architecture of these models, however, for faster development iterations, it was decided to use a pre-trained feature-extractor, and to either fine-tune it, or just build a classifier system on top of it.

The ResNet18 model, by itself has a powerful feature extractor, however just appending a classifier head on top would yield purely random results, if not further refined through specialised training for the album-recognition task.

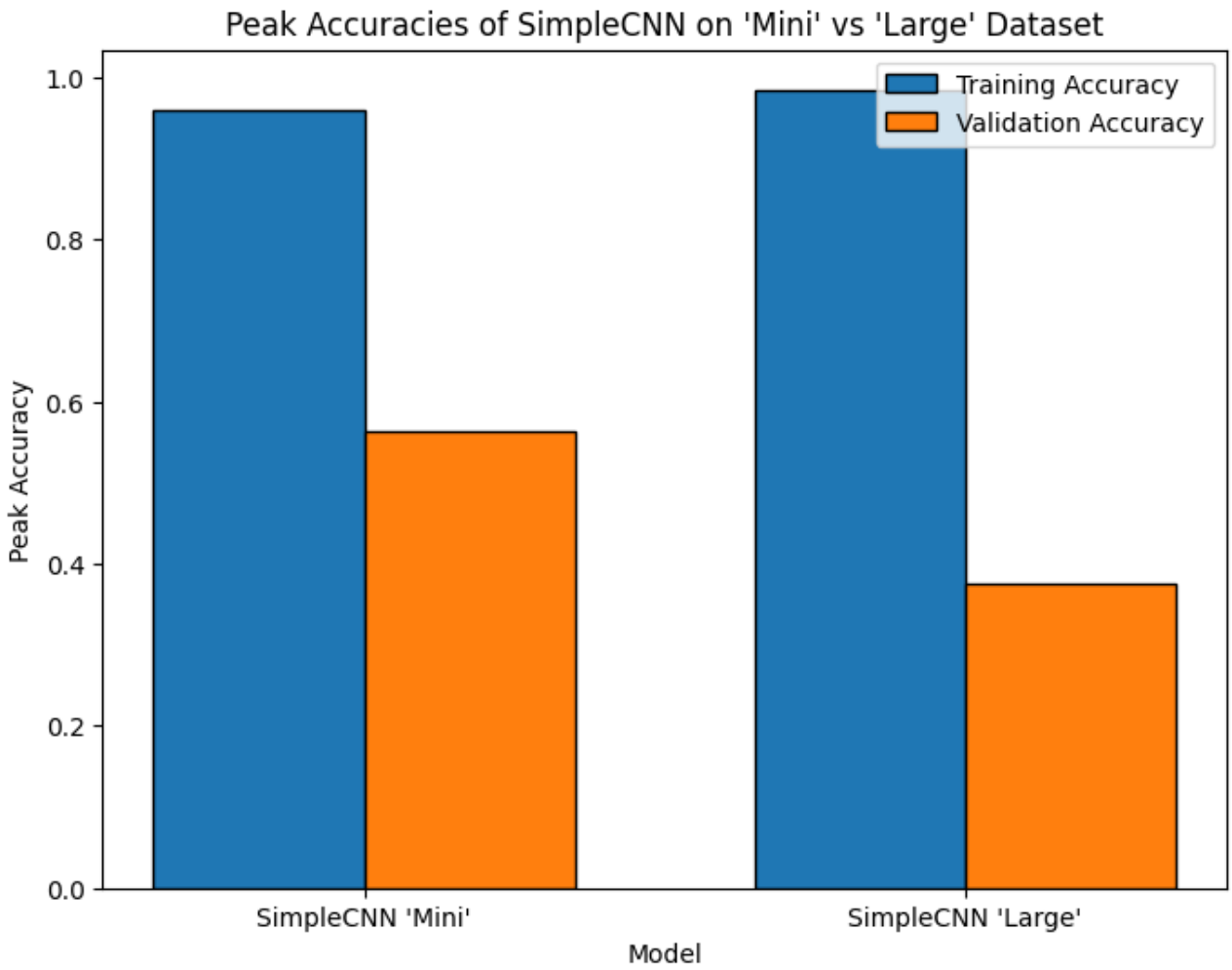

Fig. 11. Comparison of best-case SimpleCNN performance on different sized datasets

Trained on 'Mini' (D.1) and 'Large' (D.2) datasets; both validated against D.4.

4.4.2 Model Experiments

Initially, all ResNet18 weights were frozen, with only the appended classifier head being trained. This approach was intended to avoid heavy biasing and retain pre-trained knowledge, and it already outperformed the in-house DeepCNN model. Further experiments involved selectively unfreezing and fine-tuning varying numbers of layers from the base model to inspect the affect this had on performance.

As shown in Figure 12, the highest validation and test scores were achieved when fine-tuning two layers of the base model. This configuration was therefore preferred in most tests. However, while the 2-layer setup yielded the highest peak results, it also exhibited a broad and low-spread distribution across training epochs. This highlights the importance of choosing the right number of training epochs: stopping too early or training for too long can significantly degrade performance, and so particular care had to be taken in other experimental scenarios using this configuration, to ensure that epoch count was not the deciding factor influencing results.

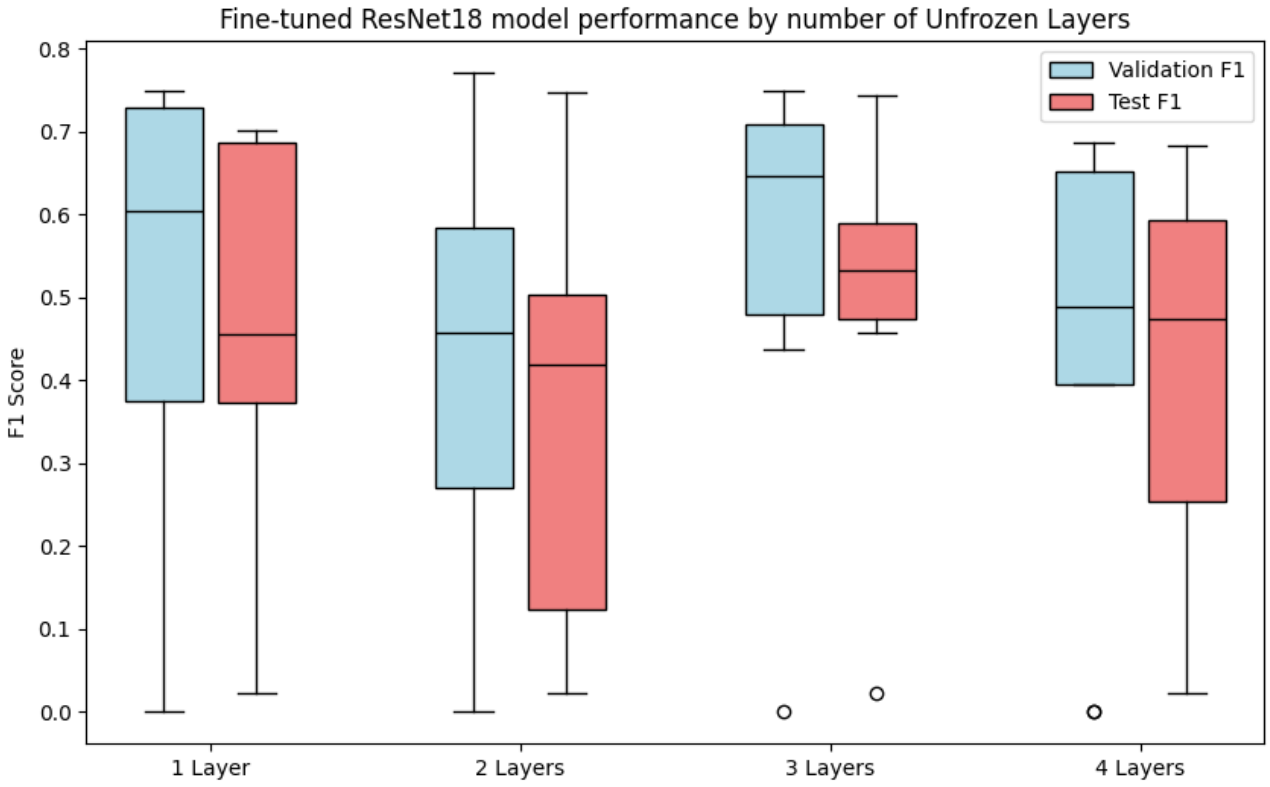

Fig. 12. Boxplot showing the distribution of F1 scores across training epochs for different numbers of unfrozen layers in a fine-tuned ResNet18.

Model Hyperparamaterisation A large-scale hyper-parameter grid search was conducted, training the model over various different combinations of value for certain aspects: learning rate, weight decay, number on unfrozen layers, batch size, etc. The results of one test, evaluating different learning rates and weight decays, over each training cycle (epoch), can been seen in Figure 13, with the best observed accuracies shown in Figure 14.

Data Augmentation In addition to experimenting with the model's configuration, the data was also investigated. Since the design phase, it was known that this model would likely not need much generalisation techniques, due to being a closed-world task, however, to make the model as robust as possible, data augmentation was still investigated. In a real use case, even though the images will be very similar to their training data, there could be slight fluctuations in rotation, size, and lighting or fading of the artwork. Additionally, some extreme transformations might occur, such as the artwork being fully upside down. In order to make the system handle this, each epoch of training was bolstered with augmented copies of with artificial transformations, de-colourings, and cropping taking place (see Figure 15 for an example).

This augmentation was generated 'on-the-fly' during training, meaning no modified image was saved or stored beyond the training epoch. Additionally, this meant that each epoch had slightly different variations it was learning on. This made the training process better for generalisation, serving as a forced dropout of seen images; however, the training process became slower, requiring more training epochs to accurately learn from the data, as well as the dataset now be scaled to be a factor bigger, increasing the time taken to train.

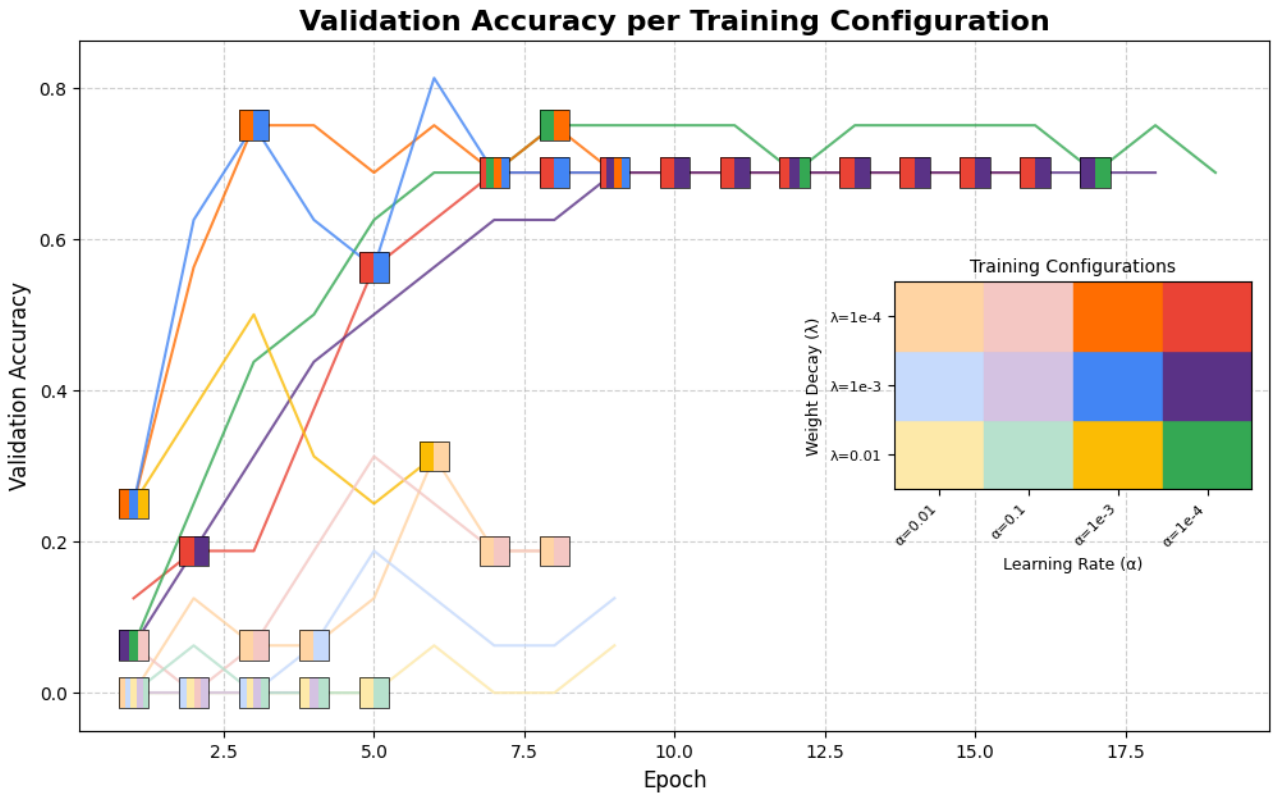

Fig. 13. Performance results of ResNet18 fine-tuned model hyperparamter grid search

Performed on D.3 training set (1,176 images) with batch size of 8; validated against D.4. Early stopping used when validation loss degraded, so not all configurations ran for equal epochs. Where multiple configurations have the same performance at the same epoch, a square marker indicates which colours are represented.

Even with GPU optimisation, as the dataset multiplied in size, it became to hold to store within the available VRAM at once, and so batch sizes had to be decreased from 32 to 8, in most cases, further slowing the training process.

This 'on-the-fly' approach also had drawbacks, as it required additional rendering computations in every epoch cycle, however, the performance gains made the extra time taken a worthwhile trade-off.

Data Hyperparamaterisation Not only was the model configuration rigorously tested under various alterations, but also the training dataset.

One notable requirement of a CNN model, is that all input data must be the same size, that is, the images must all be of uniform dimensions. The initial baseline used was 244x244, however it was theorised that greater fidelity versions of images could improve quality.

As per Figure 16, various image sizes were tested in the TODO model, to varying degrees of observed success. Whilst the 124x124 size did have the highest peak in performance, the 244x244 was most consisent in its performance. The more extreme sizes seemed to perform most poorly, likely due to the smallest images having too great of a loss in details to be learnable, whereas the larger images were likely subject to too much noise and variation, as well as potentially requiring more epochs or a deeper network with larger convolution kernels to adequately learn details.

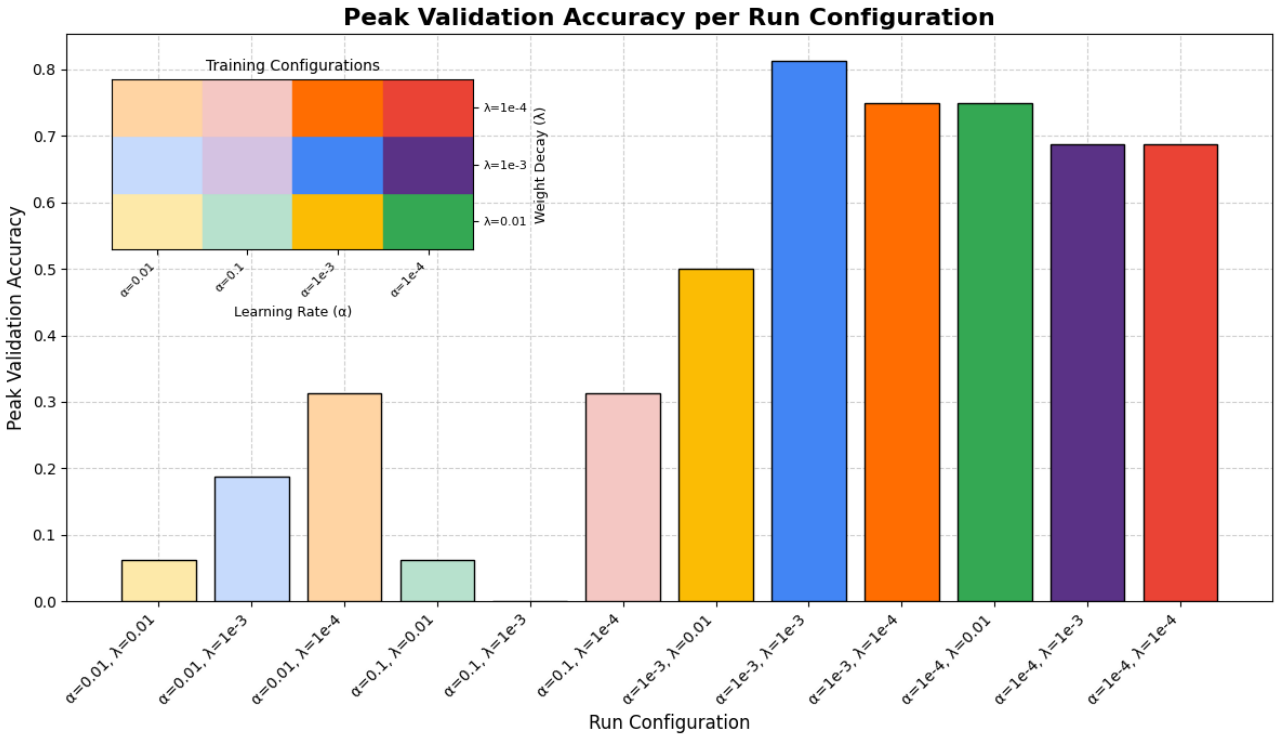

Fig. 14. Best-case performance results of ResNet18 fine-tuned model hyperparamter grid search

Full set of accuracies during training can be seen in Figure 13.

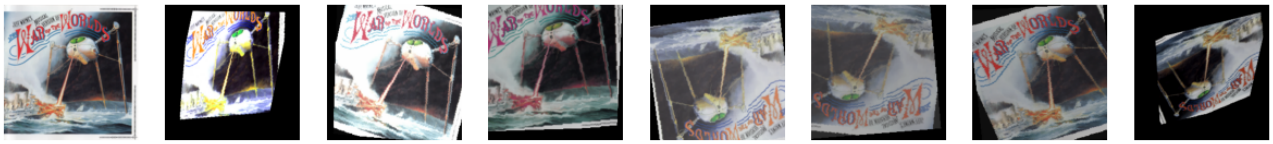

Fig. 15. Example of augmented dataset batch (without normalisation).

The first image is the unaltered original, whereas the rest of the batch have all be augmented over their rotation, size, cropping, colour, affinity and perspective.

Original artworks are © their respective copyright owners These images have been processed for research/educational purposes and do not intend to infringe upon the original copyrights.

The 124x124, 244x244, and 512x512 configurations were further experimented with, however, as per Figure 17, training time scaled roughly quadratically as the area of the image increased, an expected result seeing as convolution is highly dependent on per-pixel processes (even when done in batches, via matrix representations). For this reason, 244x244 was selected as the primary image size, due to having the most consistent performance with very small time constraints compared to the smaller 124x124 form factor.

Additionally, when switching to utilise the ResNet feature extractor, the 244x244 was especially the best-performing, as this was the input size used for the model's initial training.

4.4.3 Amphisbaena Model

The Ouroboros model, using the ResNet18 architecture, was additionally built upon to have two classification heads, allowing the model to learn different categories for the same items. In addi-

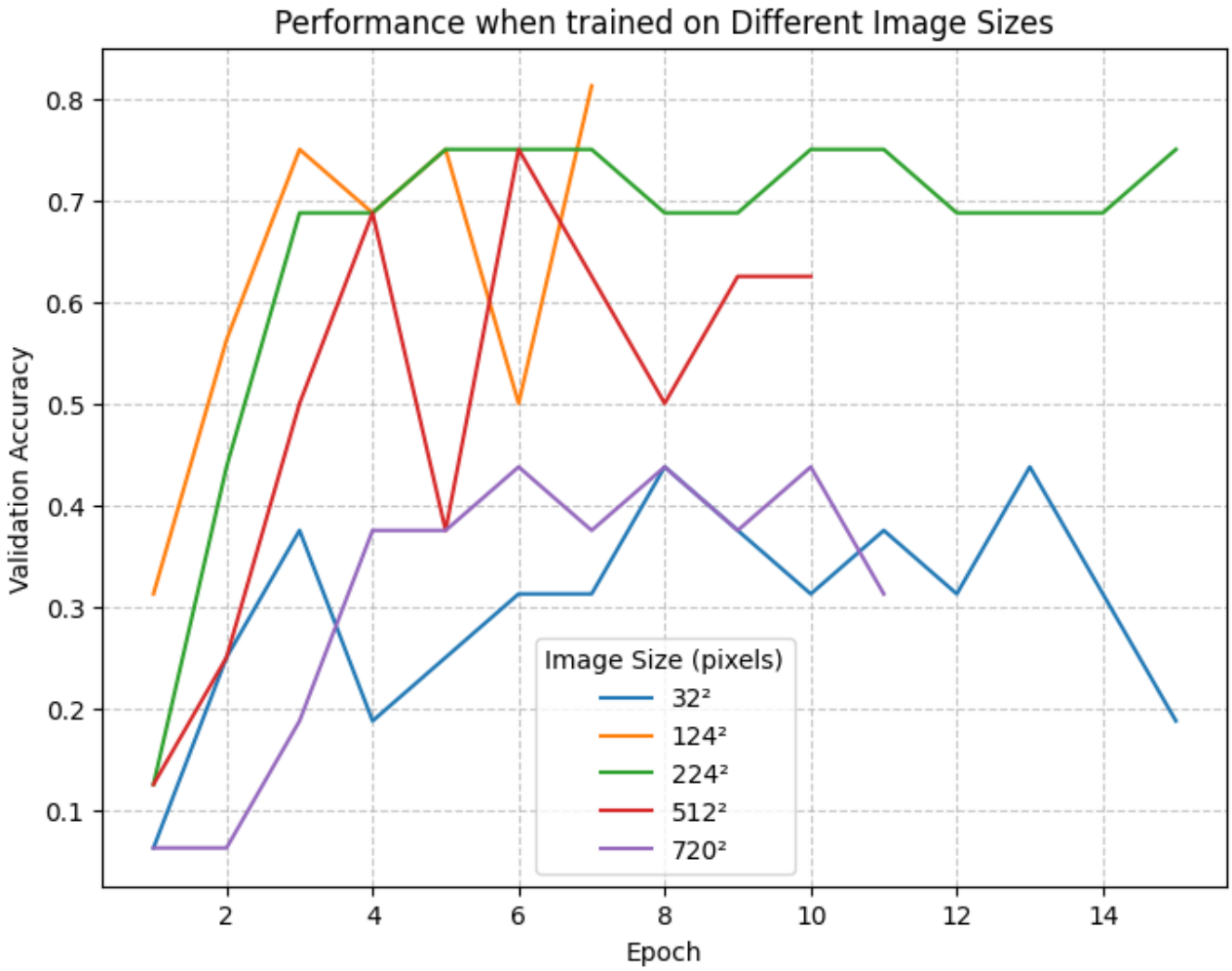

Fig. 16. TODO

Results are averaged over multiple attempts.

tion to learning the album, this allowed learning a de-coupled artist value. Using a single network, diverging into two output heads at the end allowed for shared learning from feature extraction, and reduced redundant duplication of resource-heavy processes (for example, this task could be done with two fully separate models, but would double the training time).

Multi-Head Model Diagram

This was largely done as an experimental feature, as it was unknown how well an artist can be detected, on unseen data, from other albums of the same artist. Whilst author classification is an area being explored for visual artists, this is a much more generalised task for album covers, as two albums by the same musician may feature two drastically different artworks produced by different artists, with little-to-no overlap. However, some musicians do have stylistic consistency in their albums covers. A notable example in pop culture is Ed Sheeran, whose albums often feature mathematical symbols and paint effects (see Appendix G.2, Figure 35 for a comedic example). Additionally, albums by the same musician may consistently feature a logo, brand, or even the their likeness, which could be enough information for a deep model to learn from. And, finally, many album arts are done by the same artist for a single performer or band, and so even stylistic aspects could still be beneficial.

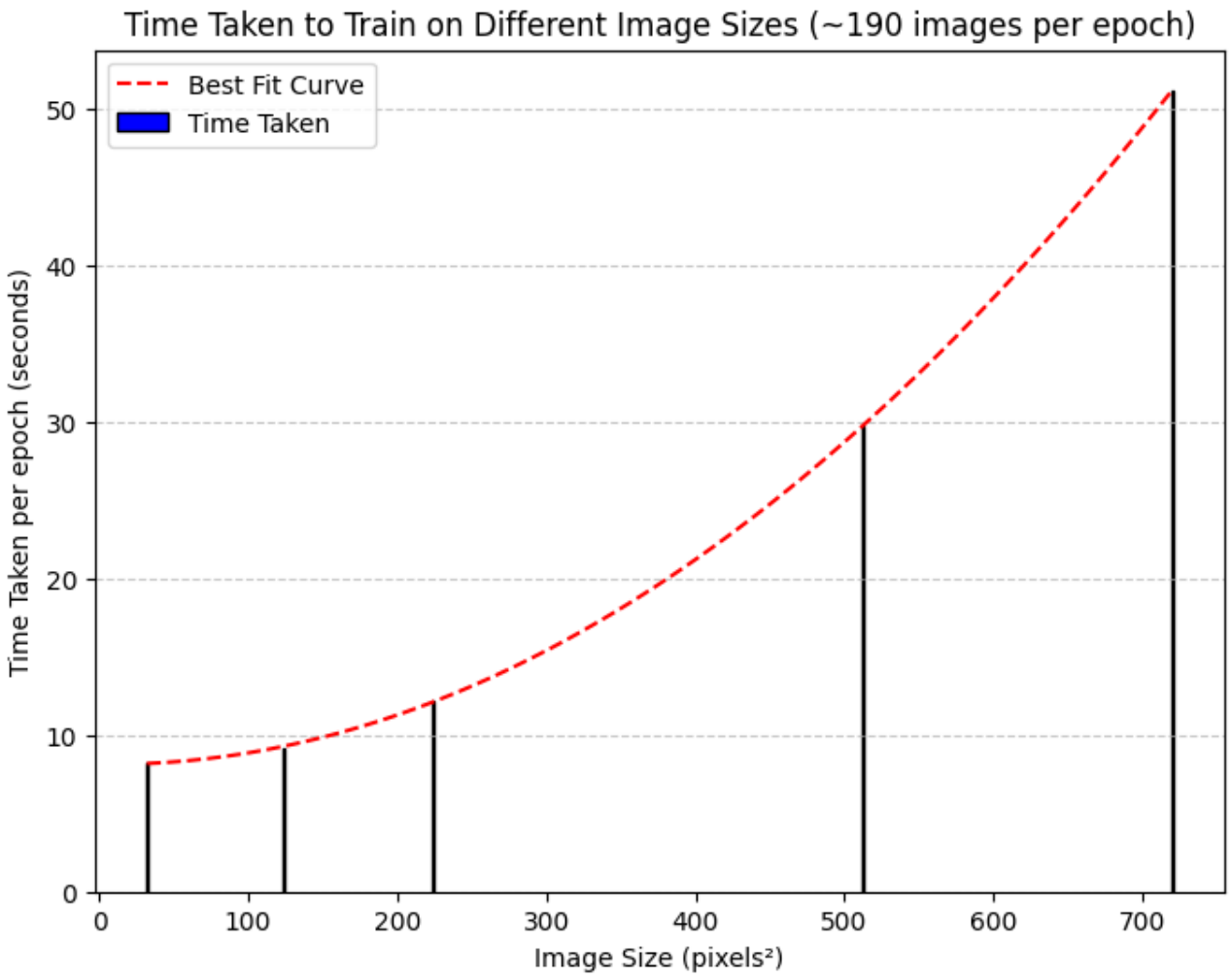

Fig. 17. TODO

Results are averaged over multiple attempts.

Fig. 18. Album covers of Ed Sheeran's studio albums

+ (2011), = (2021), × (2014), ÷ (2017), and − (2023).
Images © Asylum Records / Atlantic Records. Retrieved from Cover Art Archive.

After some experimentation, it was found that the model was able to achieve some level of success. As a proof of concept, as part of the TODO dataset, four of Ed Sheeran's albums were included. These images (the first four, from the left, in Figure 18) were able to learn enough information such that the − (2023) album (right-most image) was able to be correctly identified as an Ed Sheeran album (64% confidence).

It is worth noting that this use case is a particularly ideal one, however, with Ed Sheeran's album all being very uniformly consistent.

The aim to experiment with additional channels as classification, such as genre of decade, was ulti-

mately not viable within the time, alongside the rest of development, and so were left.

4.4.4 Hydra Model

The creation of the dynamically-expandable classification system was conducted, however, due to time constraints only minimal experimentation was conducted. The use-case for this model was that, if a user had a large collection and had set up this system to classify them, but then bought a new record, they would have to fully re-train a new model from scratch, to accommodate the new addition. Furthermore, if the model was performing poorly on some albums, the user would, theoretically, be able to inform the system of this, essentially acting as a manual annotator, which could be used to give higher weightings to these albums, to tune the model to match user expectations. This goal of high-level model customisation was challenging, however.

At first, the idea was to simply fine-tune the previously trained models, using a new collection of data. However, it was quickly found that catastrophic data loss occurred when altering weights without the original training images to validate them.

In order to avoid the retention or re-downloading of copyrighted images to be required for this system to work, instead a matrix database was created. The classification model essentially has two components: a feature extractor and a classifier. The idea was to, once initial training was complete, run each training image through the feature-extractor only, and to store the numerical tensor (the embedding) in this database. Then, if a future revision of the model was desired, the feature extractor could be frozen, then classifier could be re-initialised to allow an additional album, and the already-extracted features could be fed directly into the classifier, alongside the new data going through the full model, to ensure training saw both sets of data.

Whilst this technique showed better results, it was still prone to catastrophic data loss, either biasing towards the new data, or the original data. Freezing the feature extractor, whilst allowing to remember old features better, prevented the model from being able to correctly learn new albums; however, removing this constraint resulted in the previously extracted data sometimes being almost-random in accuracy– sometimes performing okay, but other times not; with no consistent way of ensuring ideal performance.

For this, a PNN architecture was created, allowing a dynamic number of heads to operate in parallel, using a weighted consensus approach to generate a single classification. This idea still utilised a single, frozen feature extractor, utilising the retained embeddings, however, rather than replacing the classifier, a new classifier was added in tandem to the original one. Then, during training, the two heads were weighted, through cross-entropy loss, so that the final result was the 'consensus' of the two. Essentially, the original model was preferred, unless the second model had a high degree of confidence that it was a new class, in which case, the system delegated to the newer head if, and only if, the original head was not also highly confident.

This system showed promising improvement over the early attempts, however performance degradation was still significant, with the model dropping from ~80% confidence on the initial images to ~30% (though, accuracy was maintained in the very small testing sample set used). This showed

that the dynamic approach was at least, potentially, viable for a full-scale solution. However, due to time constraints, no further experimentation was done on this concept.

A major issue with the initial vision, however, was that if a user wanted to update their model, this training system would ideally need to be run on-demand on the Pi. Due to the need of the vector database, and a deep model, for the results acquired, this means that the training process would be significantly slow, on the SBC, and therefore, might have to be done as a once-per-day system, as a limitation of the hardware.

4.4.5 Documentation

Additionally, in order to make the models clear and distinct from one another, as well as being potentially useful to other developers, a model card [42] was created, for each.

4.4.6 Challenges Encountered

On the 8th of October 2024, the Internet Archive experienced a significant cyberattack that led to a temporary outage of its services, including the Cover Art Archive. The breach exposed data from approximately 31 million user accounts, revealing email addresses, screen names, and bcrypt-hashed passwords.

This posed a fairly significant issue for the project, as the Cover Art Archive had been selected as the source of the training datasets, however, as a joint project between the Internet Archive and MusicBrainz, was among the affected services. Users reported issues accessing cover art images during the outage.

This outage lasted for weeks, with components being brought back online one-by-one, with the Cover Art Archive not being restored until after the 21st. Fortunately, a small 'toy' scale dataset had already been accumulated, in addition to the physically-owned copies that could be used, and so this did not become too much of an issue for the project. Whilst model training was limited to small-scale models, these weeks were used mostly on development of the web components.

If, however, this outage had continued for longer, then serious consideration of finding a new service would have been required. During these weeks, additional research into such systems was done, an Last.fm was selected as a backup. But, once the Cover Art Archive was restored, no further issues occurred during development, regarding data availability. Although, more outages of the service have been reported, post-development.

5 Results

What was actually produced?

5.1 Software Artefact

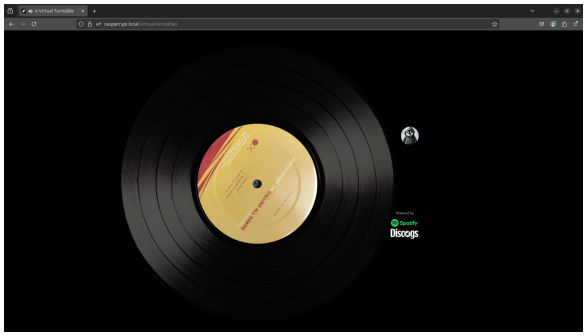

Fig. 19. Host client playing a track

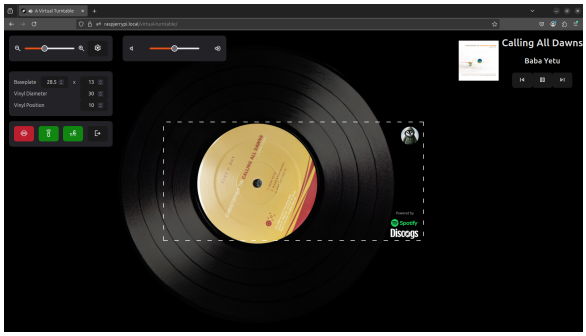

Fig. 20. Host client playing a track, with UI overlays

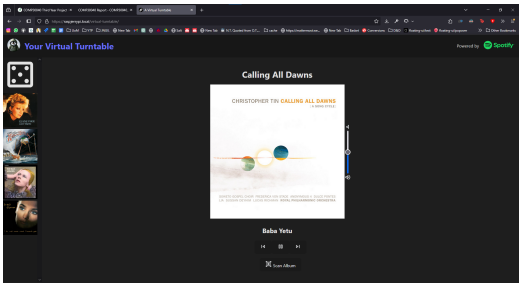

Fig. 21. Screenshot of a remote client (PC) connected to the host

5.2 Hardware Artefact

6 Evaluation

This section evaluates the system's functionality and performance across several key criteria.

6.1 Quantitative Evaluation

6.1.1 Album Classification Accuracy

To assess classification accuracy, the model was trained on the full dataset and evaluated on a held-out test set of 34 images. It achieved a top-1 accuracy of 91.18%, correctly predicting 31 out of 34 albums. Of the three errors, two were due to the model being trained on visually distinct variant covers (see Figure 25), with only one test image matching the training version. If these variant covers are treated as anomalies, the accuracy rises to 97.05%. While the test set is relatively small and not exhaustive, it reflects a reasonable physical collection a user might own.

6.1.2 Metadata Retrieval Accuracy

Since metadata retrieval depends on external sources and network responses, evaluation was performed manually. A representative set of 32 albums was tested using automated centre-level re-

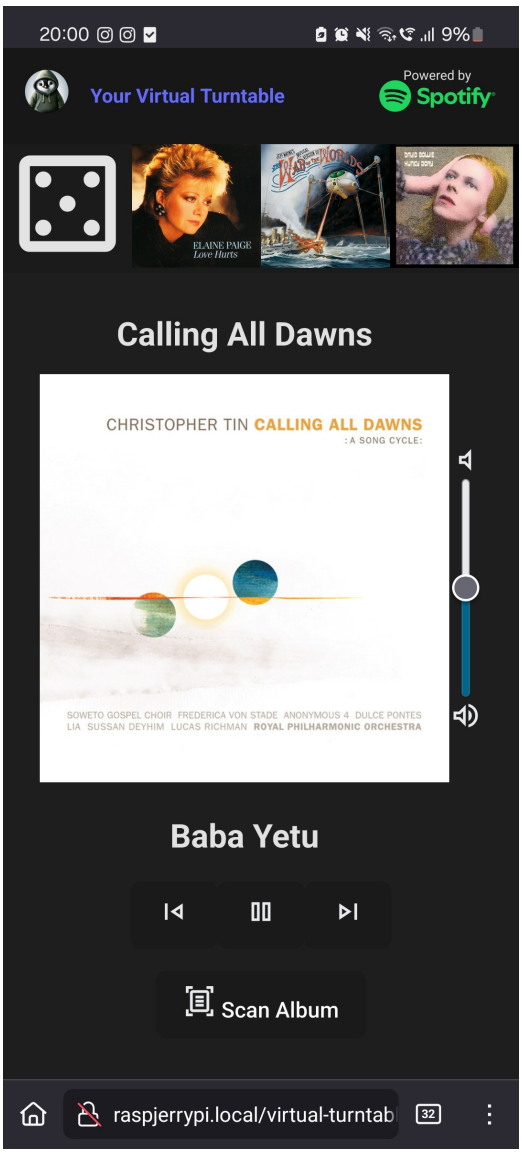

Fig. 22. Screenshot of a remote client (mobile) connected to the host

trieval. Of these, 4 were exact matches, 8 were correct but returned variant covers (e.g., CD editions), 2 matched the correct artist but wrong album, 15 returned no label, 2 returned unrelated images, and 1 returned an entirely incorrect label. In addition to the centre label, the Spotify album is also selected, which affects the textual metadata and album cover shown on remote devices. Of these same albums, 3 albums were compilations not present on Spotify, which led to retrievals of other compilations with similar themes (e.g., disco, Bond themes, Christmas). One album was a musical, correctly identified by title but with different performers, and one “best of” album was not available on Spotify, but yielded the correct artist. Despite some gaps, most queries retrieved results that were either correct or thematically relevant. With only a fairly low percentage of fetching bad data from Spotify (84.38%; up to 96.88% excluding compilations) the primary metadata source is correct, and as this only failed for particularly niche albums, a more standard collection should be very robust in the system.

The centre label system was less reliable, yielding a relevant result only 43.75% of the time, with a correct label only 37.5%; however, as this only serves additional supplementary data, this is an acceptably low success rate. Particularly as a user could add their own image samples to the sys-

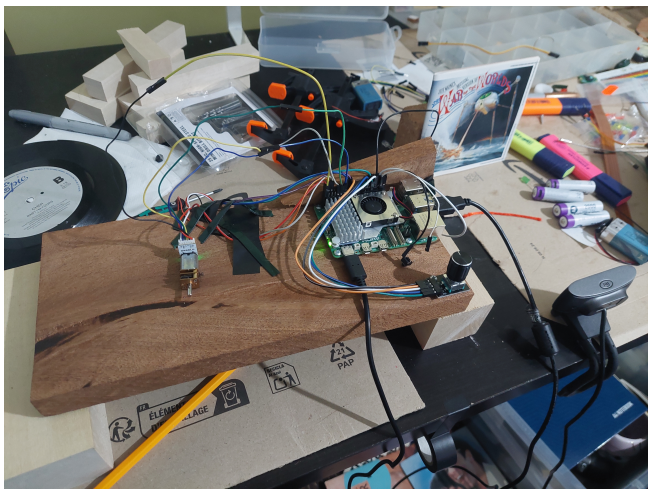

Fig. 23. Photograph of the system (scanning an album)

”Held Together By Prayers and Duck Tape” TODO: TEMP!

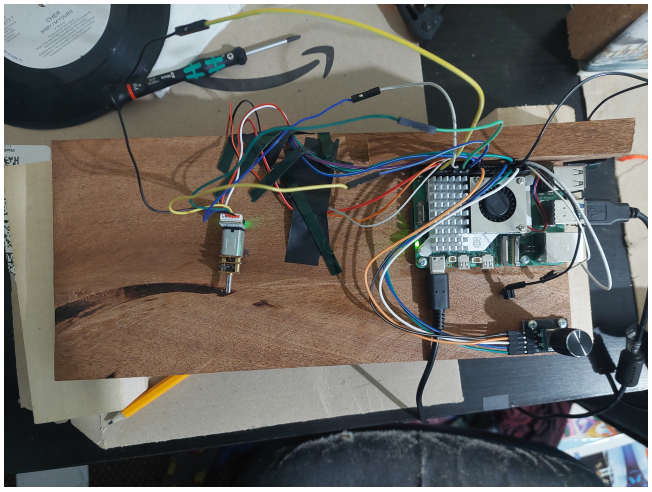

Fig. 24. Photograph of the system (top-down)

”Held Together By Prayers and Duck Tape” TODO: TEMP!

tem’s cache directory, fixing this problem. However, future improvements to this pipeline would certainly be beneficial.

6.1.3 System Responsiveness

Responsiveness was measured by timing the interval between user input (e.g., cover upload or classification trigger) and result delivery. The system was treated as an end-to-end pipeline. Tests across varying devices and network conditions showed average response times of under 2 seconds for classification and under 4 seconds for metadata retrieval, using the Ouroboros model trained on the full dataset. Worst-case latency occurred on slower connections, reaching up to 7 seconds. Overall, the system remained responsive and usable across typical usage scenarios.

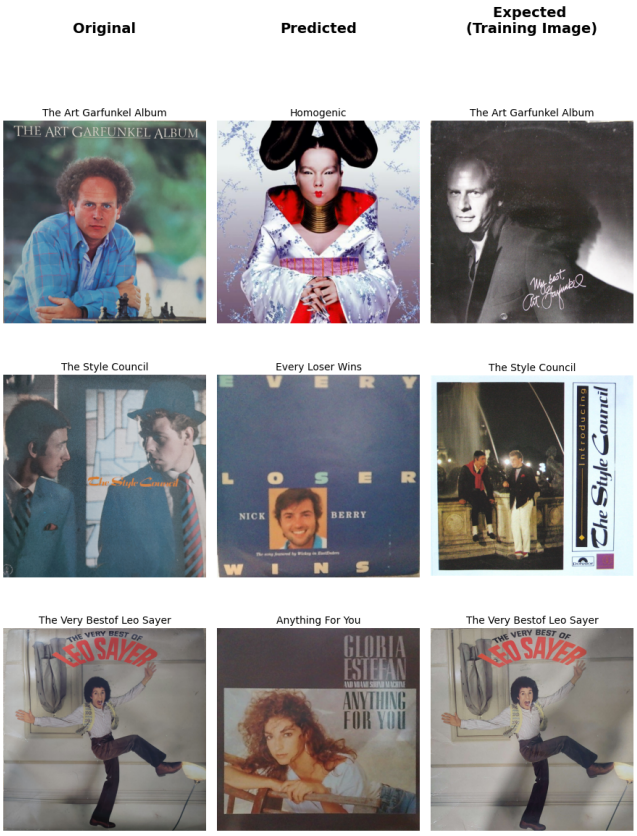

Fig. 25. Examples of misclassifications from the validation set

Each row shows a failure case: the original input image (left), the model's incorrect prediction (centre), and the correct training image (right). Errors primarily occured due to cover variants.

6.1.4 Code Robustness

The codebase was evaluated through a suite of automated unit tests, covering core components such as image processing, metadata retrieval, and classification logic. The overall test pass rate is reported, and code coverage was estimated to ensure critical paths are exercised. No critical failures or regressions were observed during testing.

The frontend achieved an average of 71.5% statement coverage, with particularly strong performance in components like WebSocketManager.ts (96%) and SpotifyPlayer.ts (92.4%). The backend reached 66.2% average coverage across core Python modules, with several key files such as sessionManager.py, stateManager.py, and modelHandler.py reaching very high coverages of 89–100%. However, some components — notably main.py, routes.py, and training utilities — remain untested due to their hardware dependence or being out of scope.

Overall, the combined project-wide coverage stands at 71.5%, reflecting a strong foundation with room for expanding test coverage to the integration and hardware layers.

6.2 Qualitative Evaluation

6.2.1 User Experience – Usability

6.2.2 User Experience – Aesthetics

6.3 Comparative Analysis

Comparison with Existing Systems

6.4 Limitations and Trade-offs

6.5 Ethical Implications

What is the impact of the finished product?

7 Conclusions and future work

7.1 Conclusions

7.2 Future work

References

[1] J. Bond, *Visualising Jesus: The appropriation and rejection of pagan divinity in pre-Constantinian images of Christ*, CLASS3200: Major Research Project, Supervisor: Samuel Gartland, 2024 (cited on pp. 12, 14).

[2] S. Bechhofer, *A virtual turntable*, Accessed: 13 Feb. 2025, 2024. [Online]. Available: `https://studentnet.cs.manchester.ac.uk/ugt/year3/project/projectbookdetails.php?projectid=55259` (cited on p. 13).

[3] H. Geraghty, "UK vinyl sales reach highest level since 1990," *NME*, Dec. 2023, Accessed: 13 Feb. 2025. [Online]. Available: `https://www.nme.com/news/music/uk-vinyl-sales-2023-reach-highest-level-since-1990-3563676` (cited on p. 13).

[4] A. Mall, “Vinyl revival,” *Journal of Popular Music Studies*, vol. 33, no. 3, pp. 73–77, Sep. 2021, ISSN: 1533-1598. DOI: 10.1525/jpms.2021.33.3.73. eprint: https://online.ucpress.edu/jpms/article-pdf/33/3/73/481613/jpms.2021.33.3.73.pdf. [Online]. Available: https://doi.org/10.1525/jpms.2021.33.3.73 (cited on pp. 13, 14).

[5] M. C. Götting. “Vinyl record buyers in the u.s. 2018-2019, by age group.” Accessed: 2024-03-02. (Jun. 2021), [Online]. Available: https://www.statista.com/statistics/1008779/vinyl-record-buyers-us-by-age/ (cited on p. 14).

[6] Guo, Yiqian, “The comeback of the medium: The history and contemporary revival of the vinyl record industry,” *SHS Web Conf.*, vol. 155, p. 02 015, 2023. DOI: 10.1051/shsconf/202315502015. [Online]. Available: https://doi.org/10.1051/shsconf/202315502015 (cited on pp. 14, 15).

[7] X. Liu, “Vinyl consumption in the age of digital music,” *Market Forum*, p. 68, 2020 (cited on p. 15).

[8] S. Hollister. “Steam now says the ’game’ you’re buying is really just a license.” Accessed: 2025-02-13. (Oct. 2024), [Online]. Available: https://www.theverge.com/2024/10/11/24267864/steam-buy-purchase-license-digital-storefront (cited on p. 15).

[9] R. Stanton. “2024 was the year gamers really started pushing back on the erosion of game ownership.” Accessed: 2025-02-13. (Dec. 2024), [Online]. Available: https://www.pcgamer.com/gaming-industry/2024-was-the-year-gamers-really-started-pushing-back-on-the-erosion-of-game-ownership/ (cited on p. 15).

[10] C. S. Legislature. “Ab 2426: Consumer protection: False advertising: Digital goods.” Accessed: 2025-02-13. (Sep. 2024), [Online]. Available: https://calmatters.digitaldemocracy.org/bills/ca_202320240ab2426 (cited on p. 15).

[11] M. McWhertor. “Warner bros. just killed a bunch of cartoon network games.” Accessed: 2025-02-13. (Dec. 2024), [Online]. Available: https://www.polygon.com/gaming/501567/cartoon-network-games-delisted-steam-nintendo (cited on p. 15).

[12] C. Bains. “The greatest lord of the rings strategy game still isn’t legally available.” Accessed: 2025-02-13. (Sep. 2022), [Online]. Available: https://www.techradar.com/news/the-greatest-lord-of-the-rings-strategy-game-still-isnt-legally-available (cited on p. 15).

[13] P. Trapp, “Only half of current vinyl buyers own a record player, study finds,” *Loudwire*, 2023. [Online]. Available: https://loudwire.com/amount-vinyl-record-buyers-actually-own-phonograph-players/ (cited on p. 15).

[14] Dictionary.com, *Audiophile*, Retrieved 13 February 2025, 2011. [Online]. Available: https://www.dictionary.com/browse/audiophile (cited on p. 15).

[15] S. Hoose, "Turning tables: Engineering the vinyl revival," *College Music Symposium*, vol. 58, no. 2, pp. 1–24, 2018, ISSN: 00695696, 2334203X. [Online]. Available: https://www.jstor.org/stable/26564886 (visited on 02/12/2025) (cited on p. 16).

[16] E. Hall, *Computer image processing and recognition*. Elsevier, 1979 (cited on p. 16).

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017, ISSN: 0001-0782. DOI: 10.1145/3065386. [Online]. Available: https://doi.org/10.1145/3065386 (cited on pp. 16, 17).

[18] C. Ramprasad, D. Saini, H. Del Carmen, *et al.*, "Text message system for the prediction of colonoscopy bowel preparation adequacy before colonoscopy: An artificial intelligence image classification algorithm based on images of stool output," *Gastro Hep Advances*, vol. 4, no. 2, p. 100 556, 2025, ISSN: 2772-5723. DOI: https://doi.org/10.1016/j.gastha.2024.09.011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S277257232400150X (cited on p. 16).

[19] S. K. Pal and A. Pal, *Pattern recognition: from classical to modern approaches*. World Scientific, 2001 (cited on p. 16).

[20] O. Russakovsky, J. Deng, H. Su, *et al.*, *Imagenet large scale visual recognition challenge*, 2015. arXiv: 1409.0575 [cs.CV]. [Online]. Available: https://arxiv.org/abs/1409.0575 (cited on p. 17).

[21] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. DOI: 10.1109/5.726791 (cited on p. 17).

[22] B. B. Traore, B. Kamsu-Foguem, and F. Tangara, "Deep convolution neural network for image recognition," *Ecological Informatics*, vol. 48, pp. 257–268, 2018, ISSN: 1574-9541. DOI: https://doi.org/10.1016/j.ecoinf.2018.10.002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1574954118302140 (cited on p. 17).

[23] G. Lin, J. Jiang, J. Bai, Y. Su, Z. Su, and H. Liu, "Frontiers and developments of data augmentation for image: From unlearnable to learnable," *Information Fusion*, vol. 114, p. 102 660, 2025, ISSN: 1566-2535. DOI: https://doi.org/10.1016/j.inffus.2024.102660. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S156625352400438X (cited on p. 17).

[24] P. Alimisis, I. Mademlis, P. Radoglou-Grammatikis, *et al.*, “Advances in diffusion models for image data augmentation: A review of methods, models, evaluation metrics and future research directions,” *Artificial Intelligence Review*, vol. 58, p. 112, 2025. DOI: 10.1007/s10462-025-11116-x. [Online]. Available: https://doi.org/10.1007/s10462-025-11116-x (cited on p. 17).

[25] H. Zheng, J. Fu, T. Mei, and J. Luo, “Learning multi-attention convolutional neural network for fine-grained image recognition,” in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 5209–5217. [Online]. Available: https://openaccess.thecvf.com/content_ICCV_2017/html/Zheng_Learning_Multi-Attention_Convolutional_ICCV_2017_paper.html (cited on p. 17).

[26] UK Government, *Copyright, designs and patents act 1988, c. 48*, Available at: https://www.legislation.gov.uk/ukpga/1988/48 [Accessed 12 Feb. 2025], 1988 (cited on pp. 18, 19).

[27] The Guardian, “UK proposes letting tech firms use copyrighted work to train AI,” *The Guardian*, 2024, Accessed 13 Feb. 2024. [Online]. Available: https://www.theguardian.com/technology/2024/dec/17/uk-proposes-letting-tech-firms-use-copyrighted-work-to-train-ai (cited on p. 19).

[28] The Times, “Christie’s faces artist protest over AI auction,” *The Times*, 2025, Accessed 13 Feb. 2024. [Online]. Available: https://www.thetimes.com/uk/arts/article/christies-artist-protest-ai-auction-artificial-intelligence-nb0hgr8mx (cited on p. 19).

[29] The Guardian, “Mass theft? Thousands of artists call for AI art auction to be cancelled,” *The Guardian*, Feb. 2025, Accessed 13 Feb. 2024. [Online]. Available: https://www.theguardian.com/technology/2025/feb/10/mass-theft-thousands-of-artists-call-for-ai-art-auction-to-be-cancelled (cited on p. 19).

[30] A. Bick, A. Blandin, and D. J. Deming, “The rapid adoption of generative AI,” National Bureau of Economic Research, Tech. Rep., 2024 (cited on p. 19).

[31] Associated Press, “Thomson Reuters wins copyright lawsuit against AI company over Westlaw content use,” *AP News*, 2025, Accessed 13 Feb. 2024. [Online]. Available: https://apnews.com/article/ai-artificial-intelligence-reuters-4a127c5b7e8bb76c84499fe12ad643c8 (cited on p. 19).

[32] H. Laddie, P. Prescott, and M. Vitoria, *The Modern Law of Copyright and Designs*, 3rd. Butterworths, 2000, Cited in *Ashdown v Telegraph Group Ltd* [2002] Ch 149, at [20.16] (cited on p. 20).

[33] Associated Press, “Paul McCartney says he fears AI will rip off artists,” *AP News*, 2025, Accessed 13 Feb. 2024. [Online]. Available: https://apnews.com/article/paul-mccartney-ai-copyright-warning-b260a4c6f0fdf732fb4994cdeb1710a4 (cited on pp. 20, 21).

[34] Spotify, *Spotify developer policy*, Effective as of 8 May, 2023, 2023. [Online]. Available: https://developer.spotify.com/policy (cited on p. 20).

[35] Spotify, *Spotify developer terms*, Version 9, effective as of 8 May, 2023, 2023. [Online]. Available: https://developer.spotify.com/terms (cited on p. 20).

[36] Discogs, *Terms of service*, Accessed 18 Nov. 2024, 2024. [Online]. Available: https://support.discogs.com/hc/en-us/articles/360009334333-Terms-of-Service (cited on p. 20).

[37] M. Heikkilä, “This artist is dominating AI-generated art. and he’s not happy about it,” *MIT Technology Review*, Sep. 2022, Accessed 13 Feb. 2024. [Online]. Available: https://www.technologyreview.com/2022/09/16/1050503/this-artist-is-dominating-ai-generated-art-and-hes-not-happy-about-it/ (cited on p. 21).

[38] The Times, “Photographer says AI being used to copy his work,” *The Times*, 2025, Accessed 13 Feb. 2024. [Online]. Available: https://www.thetimes.com/uk/technology-uk/article/photographer-says-ai-copied-his-work-can-you-spot-the-difference-q6hr5jfs2 (cited on p. 21).

[39] A. R. Skinner, “Designing for the future from the past: A modern demonstration of nostalgia through domestic interior product design,” English, Thesis, The University of North Carolina at Greensboro, 2022. [Online]. Available: http://library.uncg.edu/ (cited on p. 26).

[40] L. I. T. Attic. “The aesthetic appeal of aged brass and other vintage finishes.” Accessed: 2024-03-02. (2024), [Online]. Available: https://lookintheattic.com/blogs/news/the-aesthetic-appeal-of-aged-brass-and-other-vintage-finishes (cited on p. 26).

[41] Spotify. “Increasing the security requirements for integrating with spotify.” Accessed: 2025-02-23. (Feb. 2025), [Online]. Available: https://developer.spotify.com/blog/2025-02-12-increasing-the-security-requirements-for-integrating-with-spotify (cited on pp. 39, 40).

[42] M. Mitchell, S. Wu, A. Zaldivar, *et al.*, “Model cards for model reporting,” in *Proceedings of the Conference on Fairness, Accountability, and Transparency*, ser. FAT* ’19, ACM, Jan. 2019, pp. 220–229. DOI: 10.1145/3287560.3287596. [Online]. Available: http://dx.doi.org/10.1145/3287560.3287596 (cited on p. 53).

Appendices

A Project outline

Project outline is a required appendix. Put here.

B Risk assessment

Risk assessment is a required appendix. Put here.

C Model Cards

TODO

D Datasets

D.1 Training Set 'Mini'

15 albums; 24 data points

D.2 Training Set 'Large'

D.1

+ 68 classes; + 133 data points (a$_{dig}$)

+ 20 classes; + 39 data points (b$_{phys}$)

D.3 Training Set Augmented

D.2 * 6

D.4 Validation Set

14 albums; 16 data points

D.5 Test Set

D.4 + 1 class (_null); + 20 data points

E Evaluation Results

E.1 Unit Tests

Front End (Vitest)

```
----------------------|---------|----------|---------|---------|
File                  | % Stmts | % Branch | % Funcs | % Lines |
----------------------|---------|----------|---------|---------|
All files             |    71.5 |    81.81 |   58.71 |    71.5 |
 src                  |   76.76 |    82.08 |   47.27 |   76.76 |
  App.tsx             |   58.64 |    90.24 |   33.33 |   58.64 |
  RemoteController.tsx |   77.91 |    82.85 |      30 |   77.91 |
  VirtualTurntable.tsx |   84.65 |     75.9 |   40.74 |   84.65 |
  WebSocketManager.ts  |      96 |      100 |    90.9 |      96 |
  root.tsx            |       0 |        0 |       0 |       0 |
 src/APIs             |    42.1 |      100 |      60 |    42.1 |
  IMusicAPI.ts        |       0 |        0 |       0 |       0 |
  IMusicPlayer.ts     |    42.1 |      100 |      50 |    42.1 |
 src/APIs/Local       |       3 |        0 |       0 |       3 |
  LocalPlayer.ts      |       3 |        0 |       0 |       3 |
 src/APIs/Spotify     |   76.47 |    94.11 |   72.72 |   76.47 |
  SpotifyAPI.ts       |    66.4 |    92.85 |   66.66 |    66.4 |
  SpotifyPlayer.ts    |    92.4 |       95 |   76.92 |    92.4 |
 src/common           |   72.05 |    66.66 |   70.83 |   72.05 |
  Dialogue.tsx        |    56.6 |     62.5 |   66.66 |    56.6 |
  Tooltip.tsx         |   79.25 |    46.15 |    62.5 |   79.25 |
  WebcamCapture.tsx   |   78.78 |    86.66 |      80 |   78.78 |
 src/types            |     100 |      100 |     100 |     100 |
  Music.ts            |       0 |        0 |       0 |       0 |
  vendor.ts           |     100 |      100 |     100 |     100 |
 src/utils            |     100 |      100 |     100 |     100 |
  blob.ts             |     100 |      100 |     100 |     100 |
----------------------|---------|----------|---------|---------|
```

Back End (Pytest)

```
Name                                                Stmts   Miss  Cover
-----------------------------------------------------------------------
app\APIs\DiscogsAPI.py                                 49      2    96%
app\APIs\MusicAPI\IMusicAPI.py                         47     13    72%
app\APIs\MusicAPI\SpotifyAPI.py                       132     61    54%
app\enums\StateKeys.py                                 16      0   100%
app\main.py                                           182    182     0%
app\modules\Hardware\IHardwareController.py            31      9    71%
app\modules\Hardware\piController.py                  149    130    13%
app\modules\centreLabelHandler.py                      70     11    84%
app\modules\modelHandler.py                            70      7    90%
app\modules\sessionManager.py                          48      0   100%
app\modules\stateManager.py                            35      4    89%
app\modules\websocketHandler.py                        55     43    22%
app\routes.py                                         161    161     0%
app\utils.py                                           35      8    77%
modelling\__init__.py                                   0      0   100%
modelling\getAlbumArt.py                              131    131     0%
modelling\models\Amphisbaena.py                       127    109    14%
modelling\models\BabyOuroboros.py                      22     14    36%
modelling\models\Ouroboros.py                         107     89    17%
modelling\models\__init__.py                            0      0   100%
modelling\models\train.py                              87     87     0%
modelling\models\utils\CustomDataset.py               100     80    20%
modelling\models\utils\ModelType.py                     5      0   100%
modelling\models\utils\RandomFlip.py                   12      4    67%
modelling\models\utils\Transforms.py                    4      0   100%
modelling\models\utils\__init__.py                      0      0   100%
-----------------------------------------------------------------------
```

E.2 System Evaluation

Ouroboros Model

```
Evaluation Results:
Total images evaluated: 34
Correct predictions: 31
Accuracy: 91.18%

Failures:
TheArtGarfunkelAlbum_ArtGarfunkel_1984 --> Homogenic_Björk_1997
TheStyleCouncil_TheStyleCouncil_1983 --> EveryLoserWins_NickBerry_1986
```

TheVeryBestofLeoSayer_LeoSayer_1979 --> AnythingForYou_GloriaEstefanandMiamiSoundMachine_1988

F Image Overflow

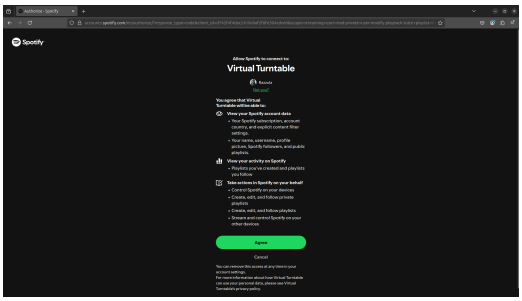

Fig. 26. Screenshot of host client using Spotify's authentication redirection flow

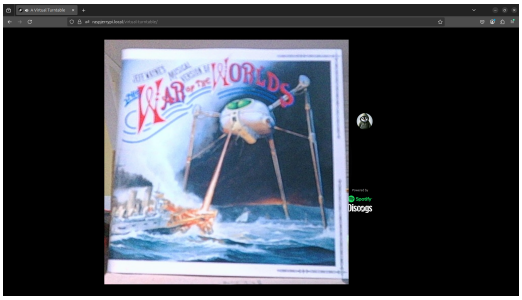

Fig. 27. Screenshot of host client displaying the input image from the camera

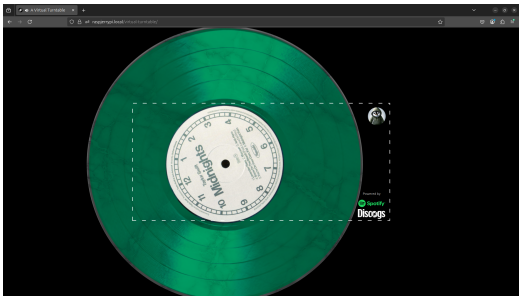

Fig. 28. Screenshot of host client using adaptive colouring and texturing

G Supplementary Information

G.1 Mythological Inspiration

Ouroboros A mythological serpent known for circling and eating its own tail, as in Figure 33. This name was used for the simple one-headed CNN model design, which 'self-fed' on deployment data with a low-distribution shift between its training data.

Amphisbaena A snake-like creature, also of origin in Greek mythology, with a notable second head at the end of its tail, as in Figure 34. This creature's name was used for the two-headed neu-

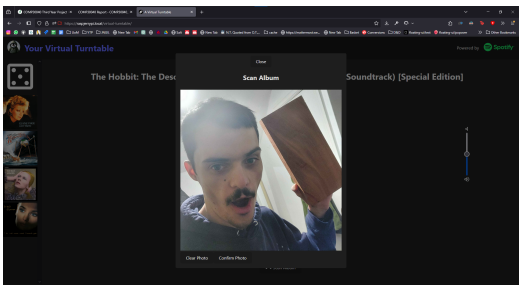

Fig. 29. Screenshot of a remote client (PC) using the camera functionality

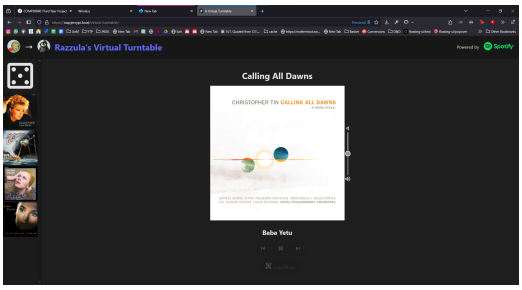

Fig. 30. Screenshot of a remote client, using an external account (not the host)

ral network design.

Hydra A rather famous mythological Greek monster, the Hydra of Lerna is a serpentine lake monster, famed for having many heads. These carying number of heads are most well-known for the fact that for every one that was cut off, two more would re-grow in its place. This creature's name was used for the PNN model design, which featured a growing number of heads.

G.2 Cultural Inspiration

Ed Sheeran Meme As a light-hearted cultural reference, a meme (Figure 35) highlighting the uniformity of Ed Sheeran's album art served partially as inspiration for designing a artist classifier. The visual similarity in album covers suggested that artist and album may be decoupled yet jointly learnable.

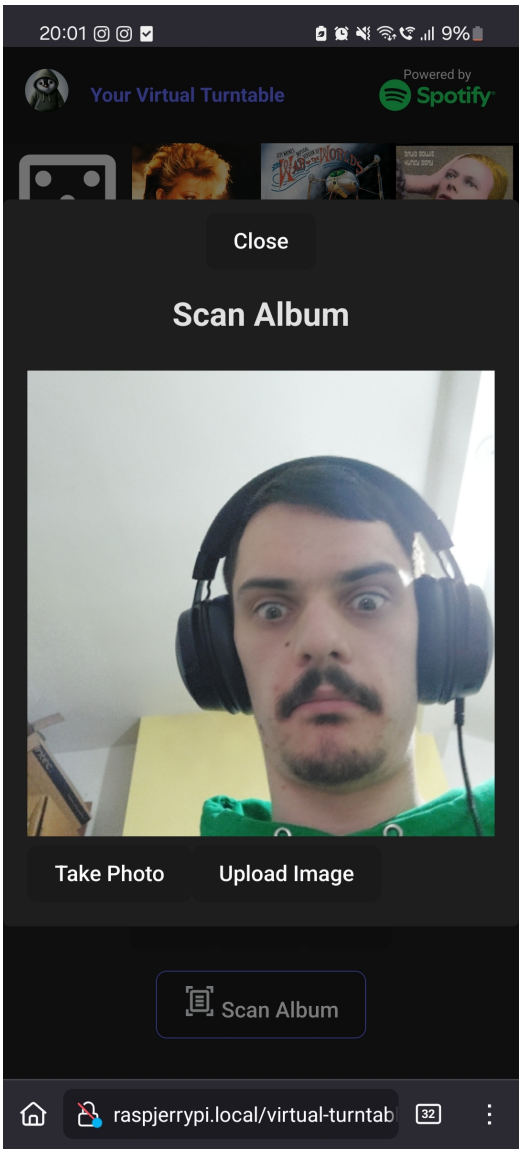

Fig. 31. Screenshot of a remote client (mobile) using the camera functionality

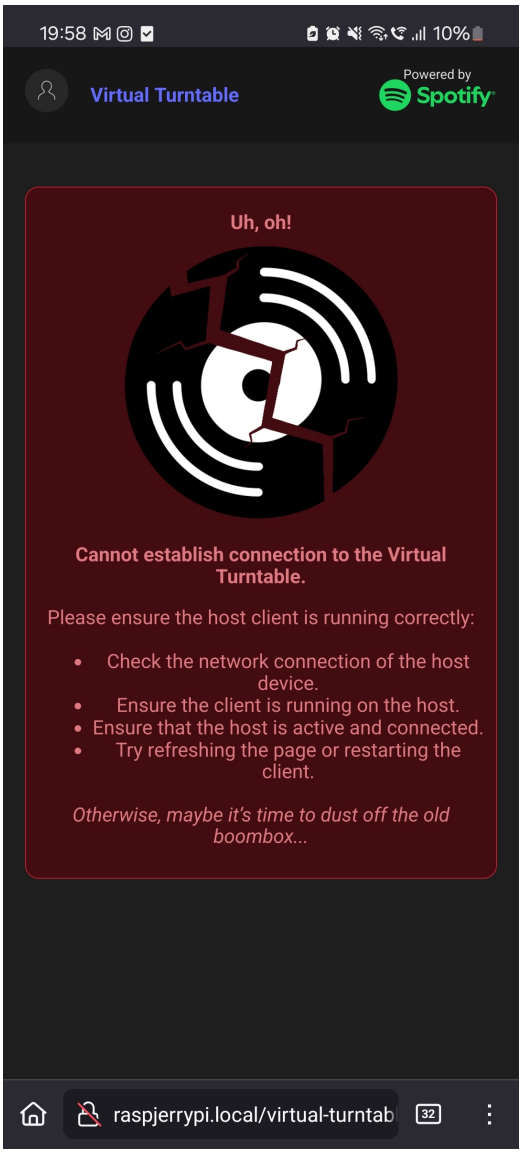

Fig. 32. Screenshot of a remote client displaying a 404 error

Fig. 33. A drawing of an ouroboros, in an alchemical tract (1478)

Source: Wikipedia

Fig. 34. An illustration of an amphisbaena (c. 1200)

Source: The Aberdeen Bestiary, folio 68V.

Fig. 35. A social media post jokingly referencing the consistent visual theme of Ed Sheeran's album covers

Source: pun_bible (threads.net)