Graduation Report

“Audio Recognition based on Machine Learning technology”

Ilyas Issatayev
452241

Table of Contents

1 Introduction

1.1 Motivation and problem indication

In 2021 the 3D animation is widely used in the media industry to create all sorts of animated movies and games. The popularity of the animation was achieved by the rapid growth of the work quality and its accessibility (PRISMARTBLOG,2019). However, the price of the development of animated movies or games stays at a high level. The average price of professional studios is " $50,000-$100,000+ Per Minute. Around $50,000 is when you can begin to expect broadcast-quality production values" (CreativeHumans, 2021). The high price of production comes from the high salaries of artists whose workflow requires them to work on all details manually.

The main focus of many media studios is to make the production of animated works cheaper and more available. One of the most common solutions is to automate some parts of animators' work to reduce their working time (FOUNDRY,2021). However, one of the innovative ways is to create animations by using motion capture (MoCap) suits. The MoCap suit allows tracking all movements of the person, the gathered data is used to make a 3D model repeat the recorded movements.

1.1.1 Motion Capture suit

Figure 1 XSENS MoCap suit

The MoCap suit contains multiple high-quality sensors that are placed on different body parts and records position and rotations relative to the world. Check the figure 1 to see how the suit looks like. The sensors have enough precision to track the movement of main body parts like hands, legs, or head, but the fingers are small in size and their movement is also smaller in scale which makes them the hardest part to record. (SkarredGhost, 2017). This issue is most

noticeable when it is required to record a musician playing on instruments such as piano or guitar. Right now sensors of the fingers are the most imprecise and expensive compared to the other parts of the MoCap suit. There are two possible ways of increasing performance quality, the first is to improve the quality of hardware sensors, and the second way is to develop software that will generate animations by using other means.

1.1.2 Musical Performance

The musical performance is the hardest stress test of the finger tracking technology due to the complexity of instrumental usage. Additionally, from 2020 to 2021, live musical performances are heavily limited or almost completely forbidden due to the COVID-19 lockdown. In order to get some revenue many artists and music labels are creating live musical performances, but online.

The artist performs in virtual reality while the audience either by using VR or just watching on screens are participating and observing. During this performance, the artist can use a MoCap suit to reflect in Virtual Reality user's body movements, which look realistic. Accurate finger tracking is an important part of creating an immersive experience. However, as it was stated before the finger tracking will be problematic.

XR Lab has an interest in finding out how the finger tracking technology can be improved.
The possible technology will provide the industry with the possibility to add value to online performances. (van Veen, 2021).

1.2 Clients

1.2.1 XR LAB

XR Lab is an organization that develops a big variety of projects in media and games spheres. It is a part of Saxion University of Applied Sciences. The XR Lab plays a role of a mediator in the current project, it has formed the developers' team and found other clients that are stated in the next paragraphs.

1.2.2 XSENS

Xsens is a company that develops full-body tracking suits and promotes motion-capture technology in the Netherlands (XSENS, 2021). It provides suits to produce 3D animation of the human body in different spheres. The focus of the company is internal sensors that can track the position and rotation of itself and share this data with the server. The suits are used to create realistic animations for games and animated videos.

1.2.3 Metropool

Metropool (Metropolis, 2021) is a company that serves as a platform for artist's performances. There are two buildings in Enschede and Hengelo that are owned by Metropool. The focus of the company is providing space, promotions, and devices for artists to perform. However due to pandemics, it is impossible now, that is why their focus is shifted to virtual concerts where they need animated 3D scenes. Metropool needs at the current moment a cheap and fast way of producing 3D virtual concerts with high quality of the art. Recording of finger position during the musical 3D performance is an important part, in order to create a hyper-realistic experience. As was explained in chapter 'Motivation and problem indication' it is hard to record finger movement, so it is important to do it differently.

1.3 Preliminary problem statement

The preliminary problem statement is formulated as follows:

MoCap suit is a finished and functional product of the XSENS company, with one of the limitations of finger tracking. The implementation of the MoCap suit that has no finger tracking hardware in live musical performances will require accurate animations of the fingers. It is therefore important to design a solution that will animate fingers during a musical performance. What technology is required to be used in order to animate hands realistically during a musical performance?

2 Theoretical framework

2.1 Research

In order to learn all aspects of the preliminary problem, the research of several related topics was done. The information was gathered from the internet and books. The research is going to be focused on finger tracking and virtual musical performances.

2.1.1 Finger Tracking

Finger Tracking is import part of the immersive performance, while the animation of body movement is precisely recorded via MoCap suit the fingers animation does not have precise tracking yet. To match the animation of the body to the animation of fingers, a lot of production power is required. That is why to prevent breaking the immersive experience of the performance, finger tracking must be done properly.

It is important to mention in the beginning that there are already existing ways of how to track fingers. There are 3 approaches to how tracking happens.

The first approach is the most straightforward, it includes sensors that are attached to the gloves. These kinds of sensors make the equipment really pricey, but give the best efficiency from all approaches. One of the most famous examples is Manus technology (MANUS, 2021) that focuses on finger tracking and they have even created special gloves to work with XSENS suits.

The second approach is to use a Machine Learning model and video camera, the Machine Learning model will detect the fingers on the video and analyse their position. The main advantage of this approach is that it almost doesn't require any money at all. The finger recognition software is available publicly online and the only thing to buy is an average camera.

The third approach is broad and can be described as “others”, this approach focuses on getting the position and rotation of the hands from any other resources and gathered data. One of the most interesting examples is the product of the JALI Research company (JALI, 2021). The company developed software that animated faces based on a speech, this product was later used in the popular game “CyberPunk 2077”. They used Machine Learning that takes as input audio of the speech and then transforms it into facial animation. The same idea could be applied to the animation of the fingers.

During first meetings with the XR Lab, Metropol and XSENS clients, it was asked the preferences of the clients in terms of the project. XR Lab suggested to use Machine Learning technology to tackle the problem. Based on Clients’ feedback and their preferences it was decided to focus more on using Machine Learning technology to solve the problem with finger tracking. That is why the most possible approach that is going to be used is the third one. In

order to understand how Machine Learning technology can be used and how to develop it the further in-depth research was done.

2.1.2 Audio related Machine Learning

"Machine learning is a form of artificial intelligence (AI) aimed at building systems that can learn from the processed data or use data to perform better. " (Oracle,2021) In simpler words, Machine Learning is an algorithm that learns based on the given data.

Machine Learning is a complex topic that has a big number of subunits that are important to research. The first step is to understand what type of Machine Learning is useful for the project. In general, Machine Learning technology is used when adaptive algorithms are needed, which is our case. An adaptive algorithm does not use a predefined set of rules as it usually was done before, but instead, it uses the logic pattern that can be applied in a completely new situation. The main advantage of the adaptive algorithm is that the developing team does not require to design behaviour for all cases.

The algorithm that is using producing output data based on Machine Learning technology is called the Machine Learning model or shortly ML model. ML model consists of the graph of nodes with weights, these nodes and weights define the behaviour of the algorithm.

The value of nodes and weights are defined during the process of "feeding" the ML model a big amount of labelled data. To increase the performance of the ML model, nodes and weights are grouped to meet the requirements of the design in the layers and these layers can be combined into a more complex ML model, this structure is usually described as a Deep Learning model. The structure of the layer and the order of layers in the model can change the behaviour of the algorithm intensely.

Different sets of layers in the ML models have their own advantages and disadvantages. Finding a suitable ML model for the needs of the project is a long process. However, in 2021 there are many pre-designed ML model types that are considered the best for certain tasks. The possible ways of picking the most suitable ML model type are by running tests and asking for help from an expert.

Once the ML model type is going to be defined the next equally important task is to find a labelled dataset to feed to the ML model. The labelled dataset is a dataset that contains a description of the data as well. As it was mentioned before the value of the nodes and weight are defined during "feeding" of the data to the ML model, this process is called training of the ML model. ML model during training trying the find patterns based on the labels and the data that was given to it. The ML model type and the amount of the labelled data are the two main variables in developing an efficient ML model.

2.1.3 Audio

ML model needs a huge amount of data to perform properly and aiming to recognize all musical instruments within one ML model is an enormous task. Hence it is important to pick one musical instrument to work with. Choosing an instrument is an important task because each instrument has its own advantages and disadvantages for developing ML models and producing animation for fingers. Based on client Metropol's feedback the 3 instruments were chosen: piano, guitar, and violin. The piano was chosen as an instrument to focus on. The piano has the advantage of being more suitable for hand animation, it is required to move hands parallel to the piano and fingers are moving on average only down when the key is pressed and up when the key is released. Additionally, piano has an advantage in the logic of usage, to play a sound you just need to press a key. Other instruments like guitar and violin require to hold strings and then execute the sound by putting a force on the string, which makes the animation of hands more complex.

2.1.4 Virtual Musical Performance

To get as much data as possible from the environment it is important to research how virtual Musical Performances are done. Fortunately, it has already become a common type of entertainment and therefore many examples can be found on the internet. The average studio that provides VR performances contains all sound recording hardware, VR trackers, MoCap suits. There are usually established two communications between artists and the audience.

2.1.5 Machine Learning Structure

Further research showed that the creation of the one ML model that takes audio as input and produces animation as an output is a too big task for the time boundaries of the project. Therefore, it was decided by the team that this task will be split into two parts: Note Recognition and Finger Animation. Note Recognition is going to be an ML model that takes input audio signals and produces Notes that were played in the audio signals. Notes that were recognized by Note Recognition are used in FInger Animation to produce the animation of hands playing the piano. Finger Animation is not going to be an ML model, but rather use predefined code. The Finger Animation part is going to be done by another developer, this project report is going to be focused on the first part, the Note Recognition powered by ML.

2.2 Final Problem Statement

Based on provided research it was chosen to use Machine Learning technology to develop the Note Recognition software that takes as an input audio file of the piano.

The main idea is to design and develop a Machine Learning model that will take as input audio signals or files and recognize Notes in the audio. The design part includes choosing what type of ML model will be used, how Layers will be structured, and what is the format of the input and output. Additionally, the design part includes looking for the suitable dataset that will be fed to the ML model. The production part includes training of the ML model, fine-tuning and evaluating. Fine-tuning is a process where the main variables of the ML model are changing in values in order to increase accuracy.

Based on project deception and gathered data the main research question was created:

“How to recognize Notes of a musical artist’s piano song during the concert, by using a Machine Learning model, without using any additional input data?”

The question may be divided into the following sub questions:

1. What type of Machine Learning is most optimal for the piano Note Recognition task?
2. What is the optimal way to structure Layers in the ML model of the piano Note Recognition task?
3. What is the best Data type that can be used to feed a Machine Learning model?
4. What are the most optimal formats for the input data and the output data?

2.3 Scope

2.3.1. Deliverables

The project will produce the functioning ML model that recognizes notes in piano play audio files. Other by-products will be the dataset and data pipeline. Data pipeline is set of scripts that will transform data from chosen dataset into readable for the ML model format.

2.3.2. Inclusions / Exclusions

The distortion or noise level must be low. The audio file must contain a recording of only piano and nothing more.

2.3.3. Assumption

The main assumption is the possibility of creating a Machine Learning model that can analyse audio input and be able to recognize notes.

2.3.4. Constraints

The main limitation is project time which is approximately 6 months. The secondary limitation is the lack of a big amount of data and hence it is required to work with public datasets or invest in generating own data but it takes time that is limited.

3 Design

3.1 Setup

The design phase of the project includes choosing ML type, software, libraries, and establishing a general direction of projects development. All work in the design part will be based on gained information that was collected during the theoretical part. The main goal of the design is to establish the general direction of project development, approaching some research sub-questions, and show the process of the idea’s evolution.

At the beginning of the design phase, the main priority was to develop a simple ML model and manage to run it. The main obstacle is to find a suitable set of software that allows to development of an ML model and deploy it properly.

3.2 Perceptilabs

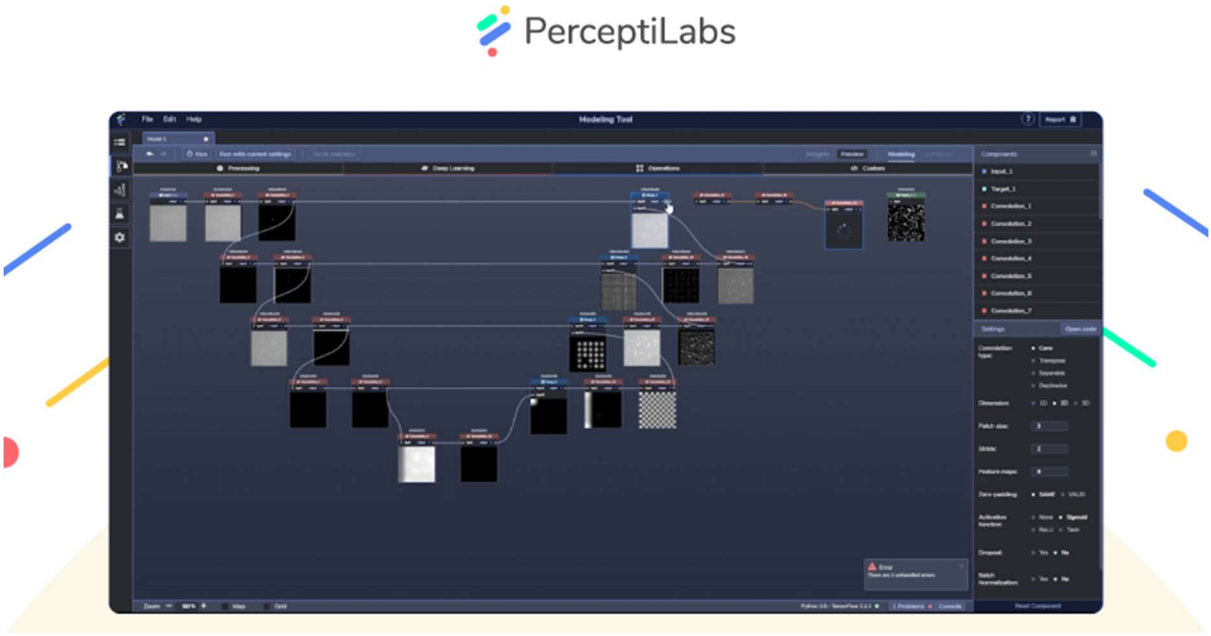

Figure 2 PerceptiLabs screenshot

Perceptilabs is a tool that helps develop and train Machine Learning models. The main advantage of the application according to our needs is a visualization of Machine Learning models and data. Check the figure 2 to see the screenshot of the software. Visualization is important for our project due to the lack of skill and experience in developing and training ML models. (Perceptilabs,2021)

Perceptilabs is a great tool to start learning Machine Learning technology. Each Layer is represented as a box that can be connected to other boxes via lines, which represents communications between layers. Additionally, the application provides extra functionality for operating with data structures (example of functionality: sorting data, merging data, changing the format of data, etc.)

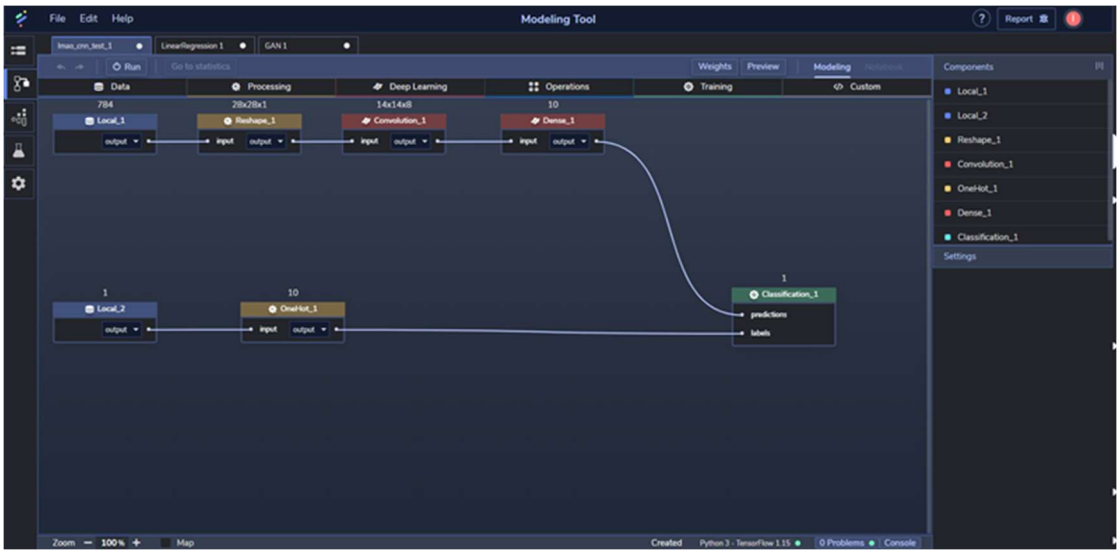

Figure 3 Screen shot from the Test

The simple ML model was developed by arranging Layers and connecting them with lines as is shown in the figure. There are two input Layers, one for the actual data and the second for the labels. Check figure 3 to see how the developed simple ML model looks like.

Labels are the metadata that represents a description of what is actual data means. For example, actual data is an audio file, and the label is a set of notes that are played in that audio file.

The first Input that contains actual data is connected to Dense Layer and Convolutional Layers. Dense and Convolutional Layers will be explained later in the report. Those layers are responsible for identifying the pattern.

At the end of the model, there is a classification layer that is responsible for the training process. Classification Layer compares results of the ML model's prediction and label and if they are not the same, the layer corrects the values of nodes and weights in the rest layers of the model. This correction of nodes and weight values is called the ML model's training.

Perceptilabs gave a great opportunity to learn simple designs of the ML models and also provided a user-friendly experience of developing ML models. However, there is one obstacle that prevented further usage of this tool in the project.

By using a small amount of data as an input (smaller than 1GB), the Perceptilabs trained model fast, but when the data size went in average above 1GB of data the Perceptilabs started to train slow.

Further investigation showed that Perceptilabs mainly uses GPU to do calculations when the ML model is training, but when the data size increases over few gigabytes it switches to CPU to do the calculations. GPU is way faster in calculating ML than CPU, that is why the speed of training is dropped.

Due to the issue with slow training of the ML model, new approaches were researched.

3.2 Tensorflow

3.2.1 First Look

Perceptilabs visualizes and helps to develop ML models, but it does not provide core functionality to run, train and deploy ML models. Instead, this tool uses a third-party library called TensorFlow, to operate with ML technology.

Tensorflow is a popular and widely supported library that has big documentation and community. Tensorflow is adapted for two programming languages: Python and C. There is a lot of extra adaptations for a variety of other programming languages, but they all have limited access to the functionality of TensorFlow.

The official website of Tensorflow states:
“TensorFlow is an end-to-end open-source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries, and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML-powered applications.” (Tensorflow,2021)

3.2.2 Technical Aspects

Perceptilabs offered development of the ML model without writing code, which caused a boost of learning progress sharply. However, Tensorflow is just a library and not an application, which means that we have to write code that is going to use the functionality of the library. Additionally, as it was stated before the programming language preferably has to be Python or C.

The most user-friendly and comfortable language to work with Machine Learning technology in 2021 is Python (Karczewski, D., 2020). That is why it was decided to use Python to write a code that can access the Tensorflow.

The code can be written in any basic application like notepad, but many applications are designed for comfortable code writing for special programming languages. These applications are called IDE, IDE is Integrated Development Environment.
Additionally, they can contain compilers that allow running the code of the given programming language.

Jupyter Notebook is an IDE that is widely used among Data Scientists and Machine Learning developers (Das. S, 2021). The main advantage of the Jupyter Notebook is that you can choose what blocks of code to run versus the normal way where the code runs from the beginning to the end.
Developing any type of code requires to always run and test the code, however some part of code are executed faster then others. Machine Learning code needs to load big amount of data, which takes a lot of time. That is why it is better to avoid running the data loading

frequently. Jupyter Notebook provides functionality of loading data once and running other part of code as many times as wanted.

It was decided to try Jupyter Notebook as the main IDE for the project.

3.2.3 Performance Test

The first test was done after python as the programming language, Jupyter as IDE, and Tensorflow as the library were chosen. The main goal of the test was to develop a simple ML model that can take as an input a big amount of data.

It is important to mention that the test solely focused on the performance of the library and was not focused on developing functioning note recognition ML.

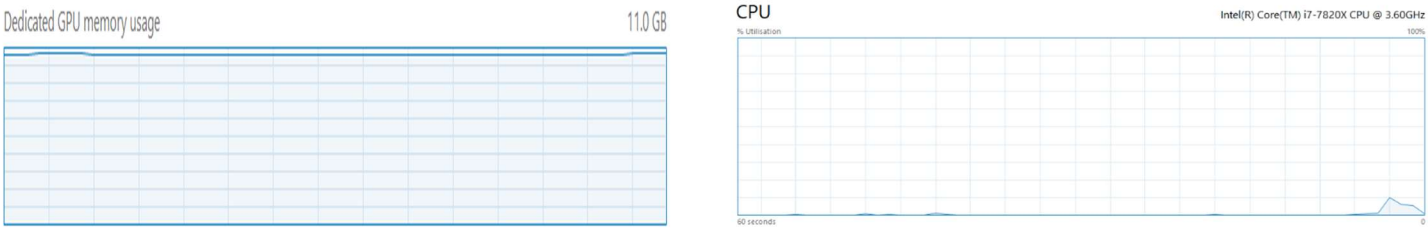

Figure 4 GPU&CPU usage graph during performance test

The ML model with 3 Dense Layers was created and 5GB of random data were fed to the ML model. During the test, the performance of GPU and CPU were tracked. Fortunately, the tracking showed minimal usage of CPU power and big usage of GPU(check figure 4). Additionally, the time of training was shorter than with Perceptilabs.

The test concluded that it is possible to develop and train ML models using Tensorflow, Jupyter, and python.

3.3 Machine Learning type

3.3.1 CNN&FNN

As it was stated in the theoretical framework many types of Machine Learning are suitable for their own specific problems. The problem that this project aims to solve is note recognition of piano, stating the problem is the first step of identifying what type of ML can be used. On figure 4 is displayed the simple ML layers structure with one input layer, one functional layer or hidden and one output layer.

According to the research paper “State-of-the-Art Model for Music Object Recognition with Deep Learning” of the authors Zhiqing Huang, Xiang Jia, and Yifan Guo, the good choice of the ML model type are Convolutional Neural Network (CNN) or Feedforward Neural Network (FNN). According to another two research paper of Arne Corvin Jaedicke “Improving

Polyphonic Piano Transcription" (Brownlee, J.,2019). CNN is the preferable choice for the ML.

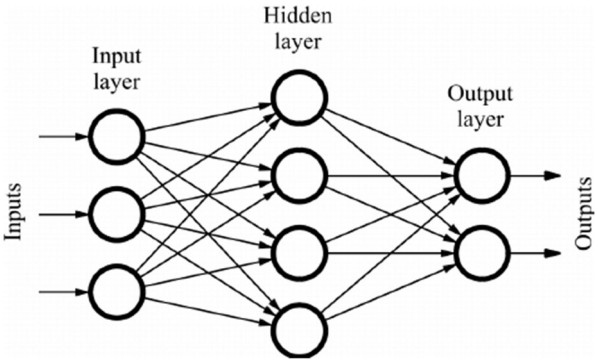

Figure 5 FNN scheme

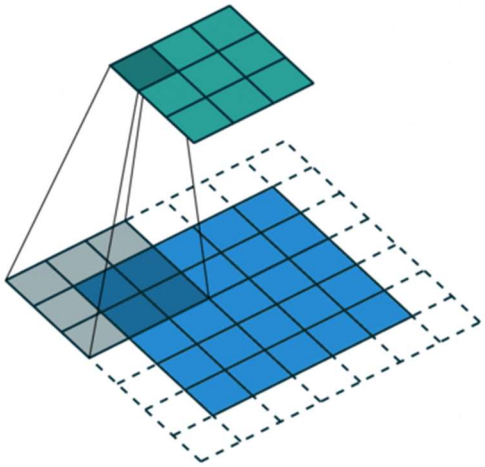

Figure 6 CNN How it works

Feedforward Neural Network is the simple Neural Network where each node in the layer is connected to each node of the next layer. FNN is suitable for any simple task where complex patterns are not present. Check figure 5 to see an example of FNN. (DeepAI, 2019).

Convolutional Neural Network is the complex Neural Network that takes as input two or three dimensional arrays. CNN is suitable for image recognition tasks, where images act as input two or three dimensional arrays. Each node in the layer is connected to all nodes around it from the previous layer, this relationship between nodes allows CNN to establish patterns and work with the visual objects. Check figure 6 to see an example of CNN (Saha.S, 2018).

Based on research papers CNN looks like the most optimal ML model type to use for note recognition. It is a complex neural network and able to establish patterns of visual objects. On the other hand, the input data of our ML model is the audio signal and not the visual picture. CNN is one of the best ML types for visual object recognition but does not provide a solution for any other formats. However, It is possible to transfer the audio files into the image.

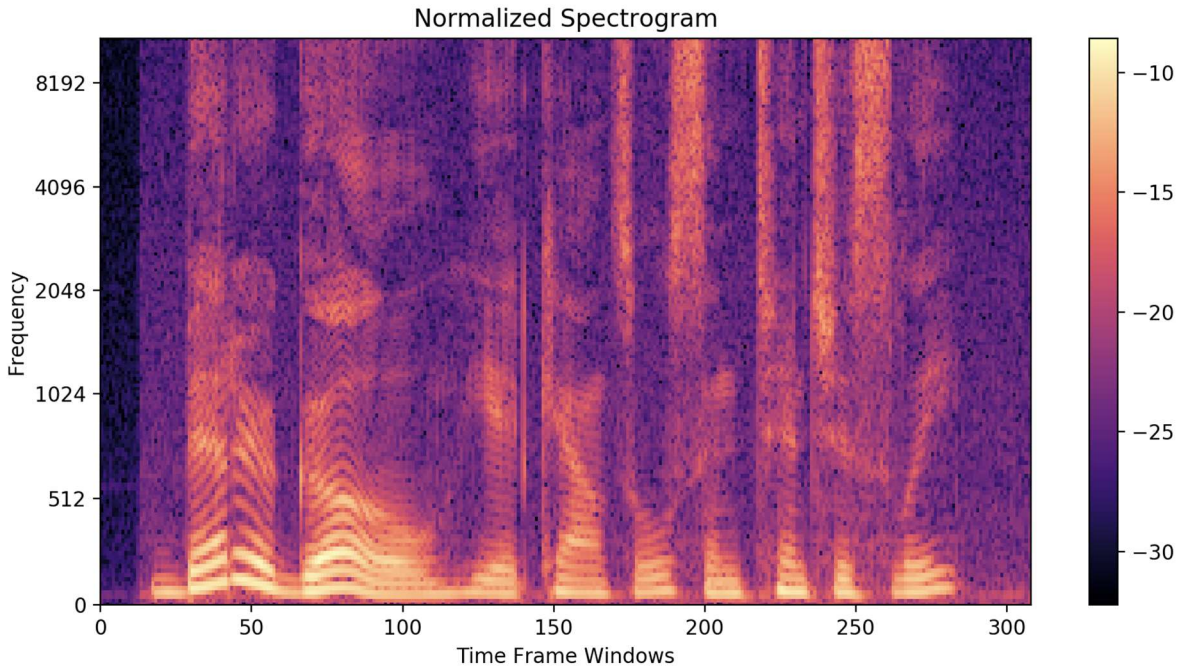

Figure 7 FFT Spectrogram

Fast Fourier Transform (FFT) algorithm can transform audio signals or files into the spectrogram which has the format of the image. The spectrogram is a graph where the y-axis is the frequency, the x-axis is time, and the colors represent amplitude. Check figure 7 to see how Spectrogram looks like. FFT spectrogram describes what frequency and amplitude were present at a specific time. As shown in the figure the visible patterns are present, wave look-alikes represent sounds and noise that are recorded on the audio file. These patterns can be detected by the CNN model. (Chaudhary, K,2021)

Now when audio files can be transformed into images, the CNN ML model type is chosen as the main ML model type. The next phase is going to be identifying the structure of the model, this process will be done in the production phase.

3.4 Dataset

3.4.1 Requirements

To successfully develop the ML model, a suitable dataset is required. The main requirements for the dataset:

- Contain sounds of notes played on a piano.
- Data is labeled
- Variety of data (all audio files must be unique)

Additionally the size of the dataset matters, the bigger is the better. The possible dataset was searched on the internet, but in case of not finding the suitable dataset, it was decided by a team that our dataset will be created. Further research was aimed at looking for a possible dataset.

3.4.2 NSynth

NSynth public dataset is developed by a company called Magenta. The dataset contains an audio file with an average length of 1.0-1.5 seconds. 11 musical instruments are present in the dataset, including piano. (Magenta,2017)

Each audio file contained 10 metadata, the figure 7 describes those metadata:

Index	ID	Description
0	bright	A large amount of high frequency content and strong upper harmonics.
1	dark	A distinct lack of high frequency content, giving a muted and bassy sound. Also sometimes described as 'Warm'.
2	distortion	Waveshaping that produces a distinctive crunchy sound and presence of many harmonics. Sometimes paired with non-harmonic noise.
3	fast_decay	Amplitude envelope of all harmonics decays substantially before the 'note-off' point at 3 seconds.
4	long_release	Amplitude envelope decays slowly after the 'note-off' point, sometimes still present at the end of the sample 4 seconds.
5	multiphonic	Presence of overtone frequencies related to more than one fundamental frequency.
6	nonlinear_env	Modulation of the sound with a distinct envelope behavior different than the monotonic decrease of the note. Can also include filter envelopes as well as dynamic envelopes.
7	percussive	A loud non-harmonic sound at note onset.
8	reverb	Room acoustics that were not able to be removed from the original sample.
9	tempo-synced	Rhythmic modulation of the sound to a fixed tempo.

Figure 8 MetaData of the NSynth

Family	Acoustic	Electronic	Synthetic	Total
Bass	200	8,387	60,368	68,955
Brass	13,760	70	0	13,830
Flute	6,572	35	2,816	9,423
Guitar	13,343	16,805	5,275	35,423
Keyboard	8,508	42,645	3,838	54,991
Mallet	27,722	5,581	1,763	35,066
Organ	176	36,401	0	36,577
Reed	14,262	76	528	14,866
String	20,510	84	0	20,594
Synth Lead	0	0	5,501	5,501
Vocal	3,925	140	6,688	10,753
Total	108,978	110,224	86,777	305,979

Figure 9 NSynth data ratio to instruments

Even so, the dataset is big, it is only one part that is suitable for the purpose of the project. Out of 11 musical instruments, it is only useful to use the piano part of the dataset, which means out of 305,979 audio samples only 54,991 audio samples can be used. Even if a small part is used it is already quite enough to develop the ML model. Check figures 8 and 9 that explain metadata of the Nsytnth and ratio of dataset according to musical instruments.

The second disadvantage is that each audio file contains only one note. It is common for a song to have chords or sets of notes played at the same time, so having multiple notes per one audio file is important. This problem can be solved by merging several audio files into one.

3.4.3 PIG

PIG dataset was built by the National Institute of Technology, Kisarazu College. The main goal of the dataset is to store piano finger position relative to the music. Due to the finger information labels of the dataset is qualitative and can help not only with note recognition but also hand animation in the future. (Nakamura, Saito, and Yoshii 2020)
PIG has 309 unique data samples, which is way less compared to the NSynth's 54,991. Additionally, all data were taken from classical piano music works, which narrows the variety of the dataset.

In general, the dataset is unique as no other dataset with finger position was found during the research. The quality of the data is high, but the quantity is too low.

3.4.4 MIDI

Musical Instrument Digital Interface (MIDI) is the format of audio files that contain data of what notes are playing in the audio file. It is not a dataset but rather a way of harvesting data from public sources. There are an uncountable amount of songs written in MIDI format, which potentially could be used as a dataset in our project. (Wreglesworth, R.,2021)

The main advantage is quantity, but on the other side, there is a big problem of keeping all data in the same form and aligned with each other. By saying aligned I mean using the same instrument, length of audio and same structure inside MIDI. It is possible to have the same song written into two different MIDI files, it is happened because of different ways how can you store notes in the file.

3.4.5 Final Choice

In the end, it was decided to choose the NSynth dataset as a main source of data. It has a big amount of data for the piano and a big amount of metadata, which makes the data more qualitative. Additionally, when a combination of audio files will be done the amount of data will be increased exponentially.

3.5 Design Results

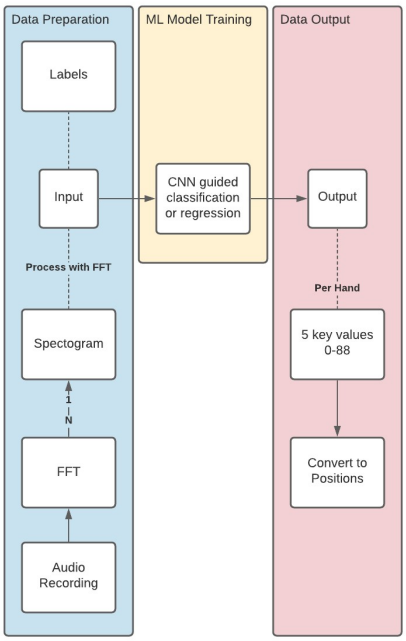

Figure 10 UML diagram of general ML structure

The figure 10 shows the UML diagram of how Machine Learning model will be trained. Data Preparation phase is mainly focused on transferring data from dataset into readable format

for the ML model. ML model training on developing the model. Data output shows how the result will be treated.

At the end of the design phase the summary of all choices is stated in the following list:

1. Tensorflow is the main ML library that is going to be used to develop the ML model.
2. Convolutional Neural Network is chosen as a main ML model type.
3. Audio files are going to be transformed into the FFT spectrograms.
4. NSynth dataset is the main dataset.

4 Production

The production phase includes establishing workflow, developing the ML model and training the ML model. Additionally, this phase includes stating the problems that occurred during the project.

4.1 Workflow

Before any development of the ML model can start, the workflow must be established. The workflow includes preparing all small pieces of the project that will help boost the progress of ML model development.

4.1.1 Hardware

The Machine Learning training process is considered pretty heavy and requires powerful computational power that GPU and CPU can provide. Additionally work with a big amount of data requires also powerful RAM and a lot of memory.

In terms of hardware, there are two possibilities, either rent a server or have own PC that can handle training. Google offers the service of renting servers for Machine Learning purposes. However, the price of the 90 days of use is 300 euros, which makes the development of the project a bit expensive.

Fortunately, the XR Lab (one of the Clients) offered a workplace in the laboratory with access to the powerful PC. After checking the performance on this PC, it appeared to be perfectly suitable for the project.

XR Lab's PC stats:

- **CPU**: Intel Core i7-7820X @3.60Hz
- **RAM**: 92 GB RAM
- **GPU**: Nvidia GeForce RTX 2080 Ti 12 GB

The XR Lab's PC was chosen as the main hardware due to flexible access and sufficiency in terms of PC power.

4.1.2 Software

There are a lot of dependencies that are needed to be downloaded to prepare for the ML development. Dependencies are the software that is required to run certain programs.

To compile python scripts, it is required to install python. The python of version 3.7 (64-bit) was downloaded and installed from their official website. (Python,2021)

There is one package manager for the python scripts and libraries that is allowing to download with just one line of command. This package manager is “PIP”. (Pypi,2021)

PIP was installed from its official website. Later on, all python related dependencies will be downloaded via PIP.

Tensorflow was downloaded and installed via PIP.

At this stage python scripts can be compiled and run, new libraries can be easily downloaded and TensorFlow libraries are already present.

4.1.3 Data Pipeline

At the beginning of the production phase, there are some assets left from the design phase such as:

- Downloaded NSynth dataset (only piano part)
- Dummy CNN model that takes as an input 2D arrays (images)

Before we start developing an ML model it is important to establish a pipeline where data from the NSynth dataset is transformed into readable 2D arrays (images) that can be fed to the ML model.

Needless to say that this transformation must be done automatically since the NSynth dataset contains 54,991audio files. Manual work will take too much time.

Python scripting is used to develop the data pipeline. The script transformed audio files of such formats as “.mp3” and “.wav” into the 2D RGB image with a resolution of 640x400. RGB stands for Red, Green, and Blue. The spectrogram was saved in NumPy format. “NumPy is the fundamental package for scientific computing in Python.” (NumPy,2020). After the script adjusted the contrast so the patterns of sound waves were highlighted more.
The data pipeline was developed in cooperation with another teammate.

4.1.4 Debugging

The Machine Learning model's training is hard to visualize and the end product is just a bunch of numbers and values. That is why it is important to build a well-working debug system to understand the outcomes of the training process. The output of the Machine Learning model is accuracy and loss values per one iteration of training. There can be more than 100 iterations. One of the solutions is simply to build a graph that has accuracy and loss line presented there over the iterations of training.

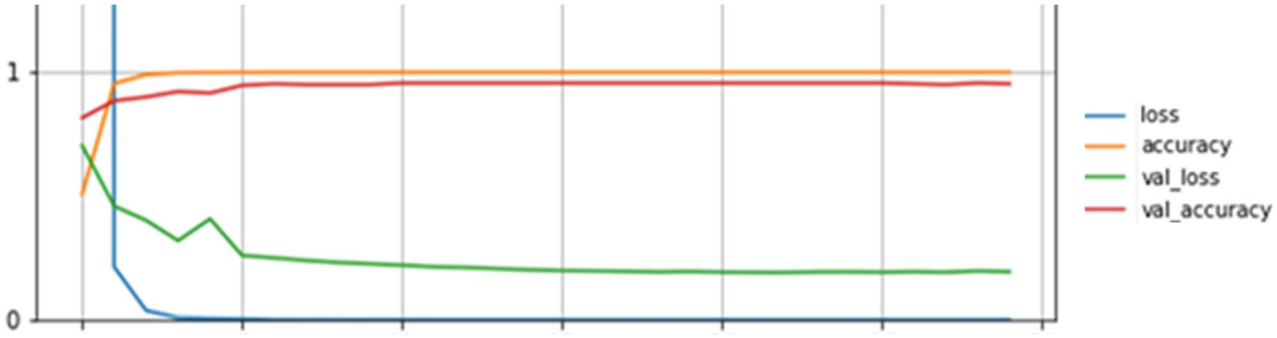

Figure 11 graph example

The figure 11 shows the accuracy and loss values over 30 iterations of training for the two sessions of ML model training.
This graph is generated after the whole training process is done and can be saved as png picture for future evaluations.

4.2 Machine Learning Model

From the design phase, it is already decided that the Convolutional Neural Network has to be developed. However there are many ways how the CNN model can be structured, CNN model means that the model focuses on Convolutional layers, the rest can be varied.

Luckily there are many well-known CNN models, but the majority of them focuses on visual recognition. During the production phase, the three CNN model structures were tested, the next paragraphs describe those 3 model structures.

4.2.1 LeNet-5

LeNet-5 is a CNN structure developed and designed by Yann LeCun et al. This CNN structure is considered the most common and one of the first-ever designed. The structure is straightforward and doesn't have many parameters. Check figure 10 to see the structure of LeNet-5.

Simplicity is the main advantage of the model's structure, which makes the training and developing processes fast. The main disadvantage is a lacking ability to analyse complex patterns or complex data, due to its limited amount of layers and feature maps. (LeCun et al. 1989)

4.2.2 AlexNet

AlexNet is a CNN developed and designed by Alex Krizehvsky, Ilya Sutskever, and Geoffrey Hinton. This model's structure has a lot of similarities with the structure of the LeNet-5 but has a lot of improvement, which makes identifying more complex patterns possible. It is the first structure that stacked convolutional layers one after another, which made it considered as a Deep Machine Learning model. Check figure 12 to see the structure of AlexNet.

Stacks of convolutional layers allow to identifying more complex patterns. However, the negative side of stacks is a high chance for the layer to become dead, which means that layer stops learning and making the training process fail.(Krizhevsky, Sutskever, and Hinton 2012)

4.2.3 ResNet

ResNet is a CNN developed and designed by Kaiming He et al. The ResNet model is more complex than previous alternatives and has a deeper structure. The main feature of the model structure is the use of "skip connections". Skip connections allow signals from previous layers not only to go through the current layer but also bypass specified layers. The main reason for creating such a complex feature is avoiding of creation of dead layers.

Dead layer is a layer that does not learn and adapt. These types of layers are preventing of data flowing to the next layer, which can damage the whole model and drastically reduce accuracy of it.

Without skip connection, the dead layer can fail the whole model, but with the use of the skip connection dead layer cannot have a big impact on the model's behaviour. The main disadvantage is its depth, which makes the training process slow and complicated. (He et al. 2015)

4.2.4 Hyperparameters

Apart from the structure of the model, several values are needed to be chosen and that have a big impact on the ML model's performance. These values are called hyperparameters. There are two main hyperparameters: loss function and optimizer function. (Brownlee, J.,2019)

The loss function's purpose is to identify how wrong is the predicted answer compared to the ground truth which represents labels. Different loss function has a different scale, but the normal format of the loss function's output is number. The loss value that we got in chapter 4.1.4 Debugging is the direct output of the loss function.

The optimizer function's purpose is to correct all layers of the model based on the grade of the loss function. Optimizer changing values of the nodes and weights to increase the accuracy of the model.

4.3 Iterations

During the production phase, 3 full iterations were done. Each iteration brought some new approaches and tackled existing problems. Additionally, unsuccessful or unfinished approaches also were stated in the report.

4.3.1 Iteration One Single Note

4.3.1.1 LeNet-5 based model

The first iteration focused on getting results as fast as possible to understand if the design phase choices were correct. The first ML model structured was chosen as LeNet-5 as the simplest to understand and fastest to train. (Check figure 12)

Layer	# filters / neurons	Filter size	Stride	Size of feature map	Activation function
Input	-	-	-	32 X 32 X 1	
Conv 1	6	5 * 5	1	28 X 28 X 6	tanh
Avg. pooling 1		2 * 2	2	14 X 14 X 6	
Conv 2	16	5 * 5	1	10 X 10 X 16	tanh
Avg. pooling 2		2 * 2	2	5 X 5 X 16	
Conv 3	120	5 * 5	1	120	tanh
Fully Connected 1	-	-	-	84	tanh
Fully Connected 2	-	-	-	10	Softmax

Figure 12 LeNet-5 structure

LeNet-5 has 5 layers excluding the input layer. The input layer had 640x400x3 dimensions, 640x400 is a dimension of the spectrogram as explained in the 4.1.3 data pipeline chapter and the last dimension 3 is there due to the RGB color code.

Data were transformed via a data pipeline and fed directly to the ML model of type Lenet-5.

For the object classification, it is usually used for the Cross-Entropy loss function, so it was used during tests. The optimizer function was Adam.

The training process runs for the 30 epochs. Epoch is an iteration of training, and 30 epochs mean that model was trained 30 times on the given data.

There are two sets of graph train stage graph which are loss and accuracy and validation stage which are

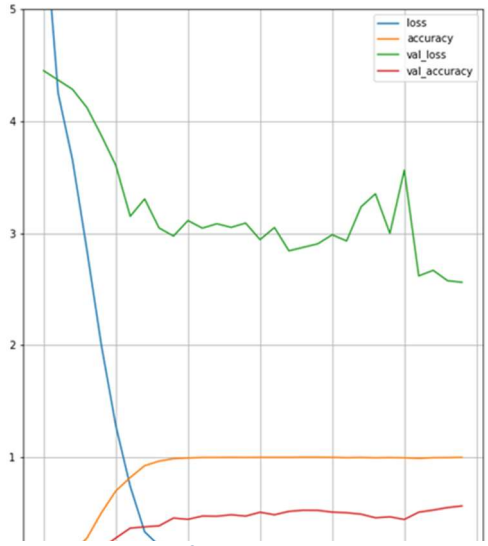

Figure 13 graph of LeNet-5

val_loss and val_accuracy. Check figure 13 to see the graph.

Validation is the process when the ML model receives as an input a small portion of the dataset that was never given to that model. The validation is done to check the performance of the ML model when completely new data is given.

The test showed that during the Train stage accuracy was 99-100% and loss value almost 0-0.1. The results for the Train stage are perfect and ideal, however the ML model already saw all these data which means it is already remembered it. That is why validation test is important to do as well. After new data has been fed to the model the accuracy dropped to 66% and the loss value rose to approximately 2.5. These results look more realistic.

4.3.1.1 AlexNet based model

The goal is to make the ML model's accuracy 100% and loss value 0, that is why a new approach was set to be tested. Due to partial success, it was decided to keep the basic structure of the LeNet-5 and improve what we have. (check figure 14)

Layer		Feature Map	Size	Kernel Size	Stride	Activation
Input	Image	1	227x227x3	-	-	-
1	Convolution	96	55 x 55 x 96	11x11	4	relu
	Max Pooling	96	27 x 27 x 96	3x3	2	relu
2	Convolution	256	27 x 27 x 256	5x5	1	relu
	Max Pooling	256	13 x 13 x 256	3x3	2	relu
3	Convolution	384	13 x 13 x 384	3x3	1	relu
4	Convolution	384	13 x 13 x 384	3x3	1	relu
5	Convolution	256	13 x 13 x 256	3x3	1	relu
	Max Pooling	256	6 x 6 x 256	3x3	2	relu
6	FC	-	9216	-	-	relu
7	FC	-	4096	-	-	relu
8	FC	-	4096	-	-	relu
Output	FC	-	1000	-	-	Softmax

Figure 14 AlexNet structure

The AexNet CNN model was examined and tested. AlexNet uses stacks of Convolutional layers on top of each other. Many training sessions were done trying different configurations to find the one with higher accuracy and smaller loss value.

In the end, the new ML model was designed and tested. Red boxes represent convolutional layers, yellow max-pooling layers and blue represent dropout layer. The last green is an output layer. Check the figure 15 to see the structure of the model.

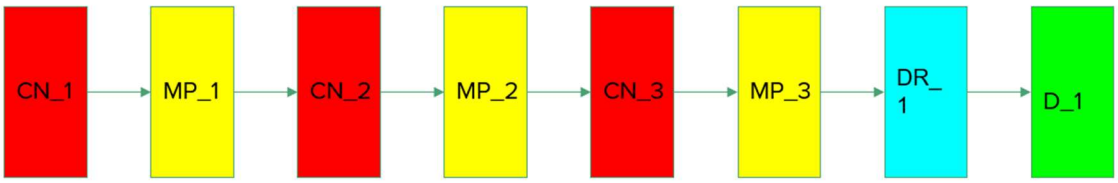

Figure 15 Custom ML model structure v0.1

It is important to mention that the logical solution would be to add more layers so the complex patterns would be easier to identify. However, the hardware that we got from XR Lab had limited GPU power. That is why the model with a high amount of layers could not be handled and trained by the PC.

The custom CNN ML model was designed and the training process was commenced. The results are shown on a graph.

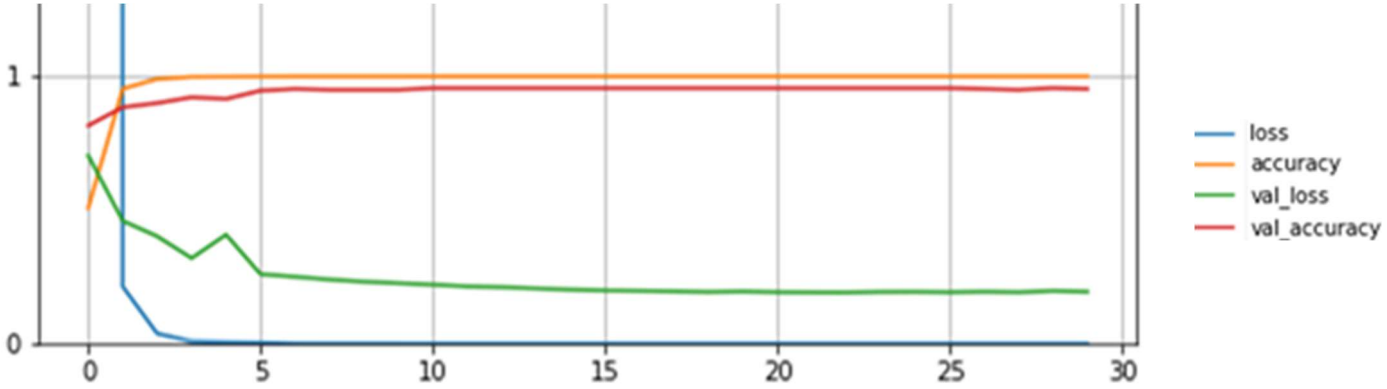

Figure 16 graph of custom ML model v0.1

The training accuracy stayed at 99-100& and the loss value completely dropped to zero levels. The validation accuracy rose to 96% and the loss value dropped to 0.1-0.2 The iteration is successful, the goal is achieved. Check the figure 16 to see the graph.

However, there is one big problem that is needed to be faced. The current ML model takes as an input audio file from the NSynth dataset and all NSynth audio files contain only one note. ML model is designed to give as an output only one note. The Note Recognition of piano audio files containing one note is done and the next step is to expand it to recognize multiple notes per time.

4.3.2 Iteration Two Multiple Notes

4.3.2.1 Switching to Multiple Notes

The first iteration showed that it is possible to do Note Recognition with high accuracy. The first issue that was faced is an unfitting dataset, the NSynth has one note per audio file, but as was explained in the design phase, it is possible to merge multiple audio files into one and then use it.

An extra python script was developed that took randomly multiple files and merged them, also saving the changes in the labels. After files were merged and labels are updated the new dataset goes through the pipeline to the ML model.

The next step is to change the output format. Currently, the output format is a number from 0 to 87, because there are 88 piano keys in the standard classical piano. This format is perfectly suitable when only one key is expected as an output.

The new output format will be redesigned to the array of 88 numbers. Each unit of the array will be filled with a probability of note in the file from 0 to 1. For example, if the unit with index 53 has a 0.8 value and the unit with index 56 has a 0.9 value then it means that keys 54 and 57 are present in the file (count of the index starts from zero, that is why numbers are shifted by one).

After minor modification was applied to the design of the ML model the training process was commenced. The results sharply went down, the accuracy was lower than 10% and the loss value went over 200 points. This can be explained by the big difference in pattern complexity. That is why a new design of the ML model was required.

4.3.2.1 New Design

The new design was created by weeks of trial and error, many training sessions were done in order to understand what is the most optimal structure of the layers. Each training session the new convolutional layer was added. However, as it was explained before the depth of the model was heavily limited by the limitations of the hardware. That is why the data pipeline has to be changed once more to fit the new design of the model.

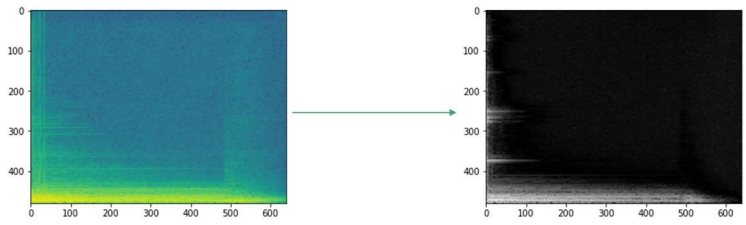

Figure 17 FFT optimization

The first change that was applied to the data pipeline was making spectrograms grey like it shown on figure 17. The main reason behind this choice is reducing the size of data fed to the ML model. The original format of the input data was 640x400x3, but after grayscale, it is reduced to 640x400. Now it is required to transfer data 3 times less than before. Figure 15, shows the grayscale transformation.

Additionally, the resolution of the image changed to 320x200. Reducing the size of data in total 6 times less than before. However, the reduction of the resolution of the spectrogram could potentially lead to losing important elements of data. The limitation of the hardware pushed to lower the quality of the spectrogram, which is an unavoidable risk that the team had to take.

This upgrade of the data pipeline allowed making deeper ML models that can handle more complex patterns. The new design of the ML model was done and it contained significantly more layers than before.

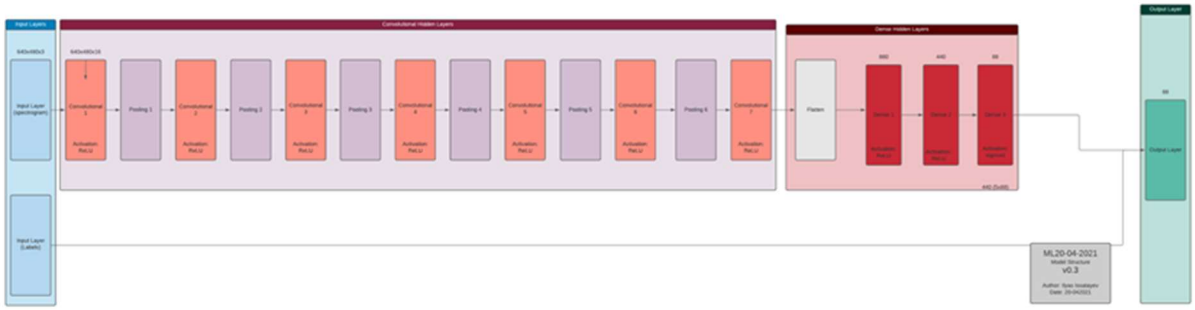

Figure 18 custom ML model structure v0.2

The new design of the ML model was constructed from the two ML model types, CNN in combination with FNN. CNN and FNN were explained in chapter 3.3.1 CNN&FNN. In the CNN part, the model contained 7 convolutional layers and 6 max pool layers. In the FNN part, the model contained 3 fully connected layers. This design proved to be the most optimal from many tries that were done before. The design displayed in figure 18.

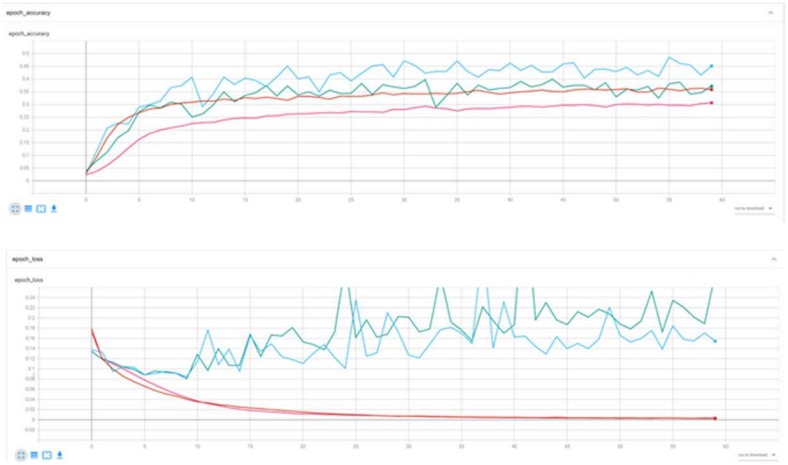

Figure 19 Graph of the ML model v0.2

The results were way lower than in the single note ML model. The accuracy at best during the training stage was 35% and during validation 45%. The loss value during the training stage was 0.02 and during evaluation 0.15. The output is shown on figure 19. A lot of time was invested in the custom ML model, but the results were unsatisfactory.

A new approach was needed to overcome the 45% accuracy barrier. It is hard to tell why exactly the accuracy was never rising, mainly due to depth of the model. There are two main factors that affect performance of the ML model, they are dataset and layer structure. Both has been optimized for the needs of the project, but unsuccessful.

4.5 Iteration Three ResNet

The next approach to tackle the problem was to redesign the ML model's structure. The current structure is the custom upgrade of the AlexNet with some elements of the LeNet-5, both of the structure are pretty old. That is why it was decided to use more recent model like ResNet. ResNet was chosen due to the probable dead layers that occur in the deep ML models. However, ResNet has complex functionality and tensorflow does not contain it, that is why it is required to write own Layer's code.

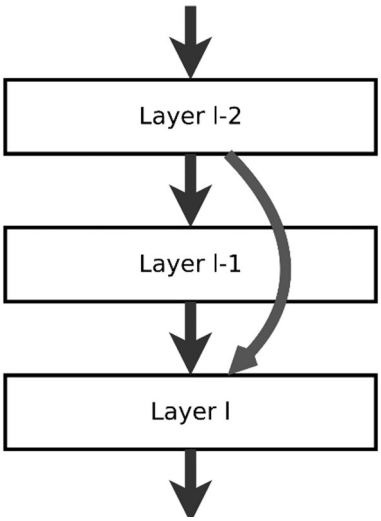

Figure 20 ResNet's skip connection

ResNet's main feature is skip connection between layers as shown on Figure 20. Tensorflow allows only direct connection between layers that is why it is required to write a custom layer. The custom layer consists of 3 layers and manages connection between them on it's own. Layer l-2 is the input layer that receives the data from previous layers. Next l-2 proceed data through own layer and then send outcome to Layer l-1 and l . Layer l-1 then takes received data and proceed it as well, next it sends proceeded data to the Layer l . Now there is an issue that Layer l received two sets of data from l-2 and l-1, so it is required to merge them. The merge process is just addition of first set of data to the second set. Later the merged data is sent to the next layers.

Multiple tests were done in order to get the highest accuracy. However, despite high hopes for the ResNet model, it did not show significant better results. It is important to mention that tests were disturbed by hardware malfunction. The main PC constantly crashed during the training process. ResNet tests were done in last month of the project, hence the time pressure and hardware malfunction caused stop of all further tests with ResNet models.

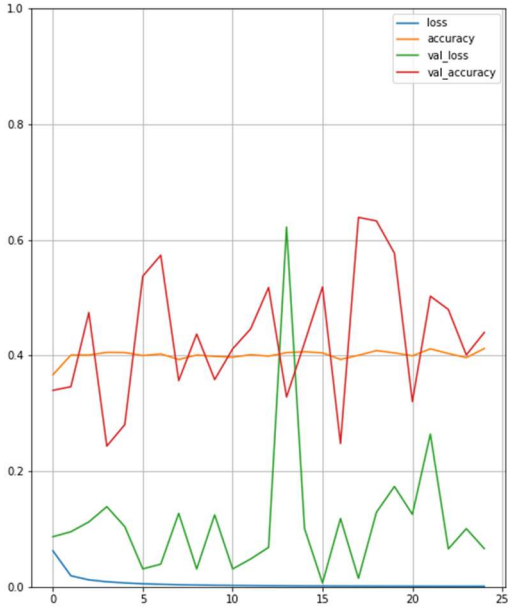

Figure 21 ResNet test Results

The most successful run of the ResNet showed that an end accuracy is around 45%, but the max accuracy was above 65%(check figure 21). Max accuracy shows that there is a potential in the usage of ResNet and further tests could potentially show better results. Even with current results ResNet has the best accuracy compared to previous models. Hence, one step closer to the efficient ML Note Recognition model.

4.6 End of Production Phase

During the production phase three big approaches were made, each of these approaches included months of many tests. With each iteration model morphed into the deeper version of itself and time of training rose proportionally. In the end of the production phase the training session lasted 6 to 8 hours. Long training time made developing of the ML model harder and longer. Despite all this each iteration brought progress and increase in accuracy.

5 Results and Suggestions

The project state by the end of the finishing deadline is going to be described in the next chapter. Additionally, problems and suggestions of what could be done differently or completely new will be added as well.

5.1 Results

The main goal of the project is to develop functional ML model that can recognise notes in the audio file. The infrastructure of software's was designed and developed in order to achieve it. Many tests were done in order to build and fine tune the ML model. The end products represent all this effort and long process of work invested in it.

By the end of the project there are many products that are done and can be use. The following list will state all of them:

1. NSynth dataset of classic piano audio files and labelled data.
2. Data pipeline that is able to convert audio files into grayscale spectrogram and parse to the ML model. Additionally, it can merge audio files with single note into one file, in order to generate audio file with multiple notes played.
3. Machine Learning model with ResNet structure that has 45% of average accuracy and 0.1 loss value. The model can recognize multiple notes with given accuracy. Input format is 640x480x1 and output format is 1x88.
4. Developer script of the same model which can be used later to train the model more or edit the layer structure. The script is important as an editable code, while the model itself is finished unchangeable product.

5.2 Problems

Despite big progress of ML model development, there were a lot of flaws that are present in the project. Majority of the flaws or problems were caused due to the big scale of the project and initial lack of knowledge and experience in ML development.
The main problem that had big impact on the project is compatibility of the hardware, CUDA & CUDN libraries and NVIDIA drivers. The right version must be installed according to the hardware and software recruitments. Frequently it wasted whole days to figure it out, which is the main reason of project slow down.
Other problems that were encountered during the project are:

- Out of Memory Error, this error occurs when the model combined with dataset are too heavy to compute for the PC.
- Long ML training sessions that can last up to 8 hours. Iterations went slower.
- Hardware malfunctions, they force to transfer project to another PC. The project size including dataset achieved over 600GB. It took about 6 hours to transfer all data to another PC.
- Lack of labelled data.

5.3 Suggestions

The end product can serve as a good basement to continue research of how ML can be used to recognize notes of polyphonic musical instruments such as piano. The most obvious suggestion is to continue tests of ResNet Structure and find the most fine-tuned version. According to the max accuracy the fine-tuned ResNet model can achieve over 65+% accuracy.

Second important step to improve existing model is to either expand Nsynth3 dataset or create own dataset that contains more data. This is the most direct way how to improve performance, but expensive time and cost wise.

Apart two main suggestions, there are some approaches that did not have a chance to be implemented.

The first one is to use K Fold cross validation. In K Fold cross validation, the data is divided into K number of subsets. Each subset then used as validation dataset and the rest is used as training dataset. This way the training process of the model can be done K number of times and then the best of the subsets can be chosen as a main. The K Fold cross validation reduces bias and provide usage of data extremely efficiently. This technology can potentially fix the problem of lack of data.

Second suggestion is to improve spectrogram generation from the audio file. There are many ways how audio can be turned in the visual graph and each of them has benefits and downsides. Potentially the better spectrogram style can let ML model find patterns faster and more accurate.

Last suggestion is to continue doing iterations and looking for the most optimal CNN model type that can increase accuracy. The project stopped at ResNet as final architecture, but further modifications are expected to do after in order to achieve perfect accuracy. The depth of model can be increased, but the hardware computational power is also needs to be expanded.

5.4 Research sub-questions

Based on knowledge and experience gained during the project it is now possible to answer the research questions that were stated in the beginning of the document. The answers will represent the choices that were done in the project and the outcome of the tests.

What type of Machine Learning is most optimal for the piano Note Recognition task?
The most optimal ML type for piano Note Recognition is Convolutional Neural Network.

What is the optimal way to structure Layers in the ML model of the piano Note Recognition task?
The ResNet is the most optimal Layer structure for the Machine Learning piano Note Recognition.

What is the best Data type that can be used to feed a Machine Learning model?
The best data type is audio files and labels as metadata.

What are the most optimal formats for the input data and the output data?
The most optimal format for the input is the FFT spectrogram of an audio file and for output is a list of note ID from 0 to 87.

“How to recognize Notes of a musical artist’s piano song during the concert, by using a Machine Learning model, without using any additional input data?”

Develop Machine Learning model that uses CNN type and has ResNet as layer structure to recognise piano notes that are played in the audio file. This model can be used to recognize notes during the concert.

6 Conclusion

In conclusion I want to summarise the progress of the project from the zero level. During the research phase I am not only gathered information related to the project, but also learned Machine Learning and python skills. Many technologies were gathered and analysed in order to create a decent design of the project. During the design phase many different software and libraries were test to find out what is the best and fast way of developing ML model. Different approaches and software showed me different aspects of ML model that was useful later in the project’s production phase.

Production phase consistent of 3 different approaches that brought increase of accuracy. I was not afraid to scratch existing working model’s structure and try completely new one. This allowed the project to grow faster and gaining a lot of important skills of ML meanwhile.

The project was dedicated on creating model of note recognition, which was accomplished, but the accuracy is still relatively low (45%). The time constrain was the only limitation that stopped from the accuracy grow after the approach with ResNet started. More test and trains session needs to done in order to make the project market-ready.

The whole data pipeline was established that transferred big amount of data each training process. This pipeline can be adapted for other future projects.

Machine Learning technology as well as python programming language were completely unknown for me in the beginning of the project but grew intensely over the whole graduation time. I have learned the general theory of ML and narrow knowledge of CNN models.
The project consisted out of 3 phases: Theory, Design and Production. Each of the phases were done properly.

The project showed big potential in using CNN models for the Audio recognition tasks. It can be used for two purposes. The first on is research, as the data pipeline with existing models and datasets can be used to make further tests and experiments. Custom layer of ResNet can be modified and reused in the future model’s structure, this will allows to build own architecture customized for the goal of the project. The second way is to increase accuracy and transform model into the market-ready product that can be used to recognize notes.

References

Unknown Author, (2021, 01 17). How much does animation cost?. From CreativeHuman: https://www.creativehumans.com/blog/how-much-animation-cost

SkarredGhost, T. (2017, 01 27). Why full body virtual reality? (and how?). From Medium: https://medium.com/@SkarredGhost/why-full-body-virtual-reality-and-how-b2cf6783be8c

Metropolis. (2021, 03 02). Metropool. From Metropool: https://metropool.nl/?gclid=Cj0KCQiA4feBBhC9ARIsABp_nbUhd2SockeQL3rxTj5xio_fn2UNcciPgH02XAv8hJ7a20xlDjH5XgaAtVREALw_wcB

XSENS. (2021, 02 15). From XSENS technology: https://www.xsens.com/?utm_term=xsens&utm_medium=ppc&utm_campaign=NWEC+%7C+Search+%7C+Brand&utm_source=adwords&hsa_cam=11538702436&hsa_src=g&hsa_mt=e&hsa_ver=3&hsa_net=adwords&hsa_tgt=kwd312964997959&hsa_acc=1306794700&hsa_grp=113335232620&hsa_kw=xsens&h

JALI . (2021, 02 17). From JALI research: http://jaliresearch.com

MANUS. (2021, 05 06). From MANUS VR company: https://www.manus-vr.com/

PRISMARTBLOG. (2019, December 30). Growth and development in animation industry. Prismart Blogs. https://www.prismartglobal.com/blogs/new-trends/growth-and-development-in-animation-industry/.

Creating an efficient animation pipeline for your studio. Foundry. (2021, April 27). https://www.foundry.com/insights/film-tv/animation-pipeline.

ORACLE. (2021). Wat is machine learning? Oracle Nederland. https://www.oracle.com/nl/data-science/machine-learning/what-is-machine-learning/.

Karczewski, D. (2020, June 18). What is the best language for machine learning in 2021? Agile Software Development Agency in Europe. Retrieved September 10, 2021, from https://www.ideamotive.co/blog/what-is-the-best-language-for-machine-learning.

Das, S. (2021, March 3). Jupyter. Analytics India Magazine. Retrieved September 10, 2021, from https://analyticsindiamag.com/why-jupyter-notebooks-are-so-popular-among-data-scientists/.

Saha, S. (2018, December 17). A comprehensive guide to convolutional neural networks - the eli5 way. Medium. Retrieved September 11, 2021, from https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

DeepAI. (2019, May 17). Feed forward neural network. DeepAI. Retrieved September 11, 2021, from https://deepai.org/machine-learning-glossary-and-terms/feed-forward-neural-network.

Perceptilabs. (2021). PerceptiLabs. Retrieved September 13, 2021, from https://www.perceptilabs.com/.

Tensorflow. (2021). Tensorflow. TensorFlow. Retrieved September 13, 2021, from https://www.tensorflow.org/.

Chaudhary, K. (2021, June 4). Understanding audio Data, Fourier TRANSFORM, FFT, spectrogram and speech recognition. FFT. Retrieved September 13, 2021, from https://towardsdatascience.com/understanding-audio-data-fourier-transform-fft-spectrogram-and-speech-recognition-a4072d228520.

Magenta. (2017, April 5). The nsynth dataset. Magenta. Retrieved September 13, 2021, from https://magenta.tensorflow.org/datasets/nsynth.

Eita Nakamura, Yasuyuki Saito, and Kazuyoshi Yoshii. “Statistical learning and estimation of piano fingering”. In: Information Sciences 517 (2020), pp. 68–85.

Wreglesworth, R. (2021, July 18). A beginner's guide To MIDI: What is It? How does it work? Musician's HQ. Retrieved September 13, 2021, from https://musicianshq.com/a-beginners-guide-to-midi/.
Python. (2021.). Welcome to python.org. Python.org. Retrieved September 13, 2021, from https://www.python.org/.

PyPi. (2021). pip. PyPI. Retrieved September 13, 2021, from https://pypi.org/project/pip/.
Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: Advances in neural information processing systems 25 (2012), pp. 1097–1105.

Alastair Lansley et al. “Caliko: An inverse kinematics software library implementation of the FABRIK algorithm”. In: Journal of Open Research Software 4.1 (2016).

Yann LeCun et al. “Generalization and network design strategies”. In: Connectionism in perspective 19 (1989), pp. 143–155.

Brownlee, J. (2019, June 16). What is the difference between a parameter and A hyperparameter? Machine Learning Mastery. Retrieved September 13, 2021, from https://machinelearningmastery.com/difference-between-a-parameter-and-a-hyperparameter/.

Arne Corvin (2019, June) Arne Corvin Jaedicke, T. U. B. (2019). https://www2.ak.tu-berlin.de/~akgroup/ak_pub/abschlussarbeiten/2019/Jaedicke_MasA.pdf. -.

Figure References

Figure 1, https://www.xsens.com/motion-capture.
Figure 2, https://www.perceptilabs.com/
Figure 3, own screenshot
Figure 4, own screenshot
Figure 5, https://www.researchgate.net/figure/A-simple-three-layered-feedforward-neural-network-FNN-comprised-of-a-input-layer-a_fig3_285164623
Figure 6, https://datascience.stackexchange.com/questions/91126/understanding-scipy-signal-convolve2d-full-convolution-and-backpropagation-betwe
Figure 7, own screenshot
Figure 8, https://magenta.tensorflow.org/datasets/nsynth
Figure 9, https://magenta.tensorflow.org/datasets/nsynth
Figure 10, own screenshot
Figure 11, own screenshot
Figure 12, https://www.datasciencecentral.com/profiles/blogs/lenet-5-a-classic-cnn-architecture
Figure 13, own screenshot
Figure 14, https://www.datasciencecentral.com/profiles/blogs/alexnet-implementation-using-keras
Figure 15, own screenshot
Figure 16, own screenshot
Figure 17, own screenshot
Figure 18, own screenshot
Figure 19, own screenshot
Figure 20, https://en.wikipedia.org/wiki/Residual_neural_network#/media/File:ResNets.svg
Figure 21, own screenshot