UNIVERSITA' POLITECNICA DELLE MARCHE

FACOLTA' DI INGEGNERIA

Corso di Laurea magistrale in Ingegneria Meccanica

BatVision and depth prediction by listening and seeing

BatVision e predizione della profondità attraverso udito e vista

Relatore:

Prof. David Scaradozzi

Tesi di Laurea di:

Elisa Corradini

Correlatore:

Prof. Roland Siegwart

A. Y. 2021/2022

In collaboration with the Autonomous Systems Lab

at the Eidgenössische Technische Hochschule Zürich

Abstract (English version)

Bats have always been considered blind. Beyond the myth, the reality lies in the middle. There are some species whose echolocation ability is complemented by the use of sight. Hence, the question that drives this research: can we see like bats, perceiving the environments and deducing distances through the integration of biological sonar and vision? Starting from the myth and the implementation of echolocation itself, we move on to the integration of the two skills, basing the work on the application of real-world data, acquired by means of sensors in apartment rooms, and on the investigation of different architectures of artificial neural networks.

Code and models are available at *https://github.com/Elisa-code184/BatVision-and-depth-prediction-by-listening-and-seeing.git*

Abstract (Italian version)

I pipistrelli sono sempre stati considerati come non vedenti. Oltre il mito, la realtà sta nel mezzo. Esistono delle specie la cui abilità di ecolocalizzazione è completata dall'uso della vista. Da qui la domanda che guida questa ricerca: si può vedere come i pipistrelli, percependo gli spazi e deducendo distanze attraverso l'integrazione di sonar biologico e vista? Partendo dal mito e dall'implementazione della sola ecolocalizzazione, si passa in seguito all'integrazione delle due abilità, fondando il lavoro sull'applicazione di dati reali, acquisiti per mezzo di sensori in stanze di appartamento, e sull'investigazione di diverse architetture di reti neurali artificiali.

Codice e modelli sono disponibili presso *https://github.com/Elisa-code184/BatVision-and-depth-prediction-by-listening-and-seeing.git*

Abstract (Italian extended version)

I pipistrelli sono sempre stati considerati come non vedenti e sono riconosciuti tra gli animali che applicano l'ecolocalizzazione per orientarsi nel mondo che li circonda. L'ecolocalizzazione, definita anche come *sonar biologico*, è l'abilità di percepire la presenza e la distanza di oggetti, come prede o ostacoli, tramite l'emissione di un segnale acustico e la ricezione del segnale riflesso dagli stessi oggetti.

Oltre il mito, la realtà sta nel mezzo. Esistono delle specie di pipistrelli la cui abilità di ecolocalizzazione è completata dall'uso della vista, decidendo a quale senso affidarsi a seconda dello scopo o integrando i due stimoli. Da qui la domanda che guida questa tesi: è possibile vedere come i pipistrelli, percependo gli spazi e predicendo distanze attraverso l'integrazione di sonar biologico e vista? In analogia con i pipistrelli, si sono sviluppati diversi approcci al tema: alcune ricerche si focalizzano sull'uso dell'udito, altre sulla vista ed altre ancora sulla fusione dei due sensi. Questo lavoro si pone l'obiettivo di trattare entrambe prima e terza opzione.

Lo scopo della tesi è quindi verificare e valutare i risultati di mappatura della profondità per mezzo dell'emissione di un suono e della contemporanea acquisizione audio e video in ambienti chiusi, come stanze di appartamento ammobiliate, in condizione di quiete e per mezzo di machine learning. Un'esemplificazione di quanto detto può essere visualizzata in Figura 1.

Con l'obiettivo di sviluppare un sistema che possa avere reali applicazioni, per esempio in ambito di navigazione robotica, la necessità di lavorare con dati reali è stata soddisfatta dall'acquisizione di un dataset adeguato allo scopo, dato che lo stato dell'arte non ne dispone ancora.

Questo documento si sviluppa quindi in tre passi fondamentali: partendo dallo sviluppo di un dataset, si prosegue con il processamento dei dati per poi passare all'implementazione ed al test di diverse architetture di reti neurali.

Relativamente al primo step, in seguito alla descrizione di sensori (Kinect V2, microfoni e cassa acustica, rappresentazione nella Figure 4) e scenari (alcuni esempi in Figura 5) adeguati ai fini dell'acquisizione, si presenta l'algoritmo (Paragrafo 2.1.4), in linguaggio Python, che ha permesso di ottenere una sequenza di campioni, raccolti in maniera concomitante alla riproduzione di un chirp di tre millisecondi, modulato nelle frequenze dell'udibile (descritto nel Paragrafo 2.1.3). Ogni gruppo di dati è composto da un'immagine a colori, un'immagine di profondità e due registrazioni audio, provenienti da microfoni distinti, che contengono il suono emesso dalla cassa acustica e il corrispondente riverbero nella stanza.

Segue la spiegazione del processamento dei dati, grazie al quale essi vengono resi input adeguati alle applicazioni di machine learning successive (Paragrafo 2.2).

Nell'investigazione delle reti neurali il filo conduttore è una struttura che si rifà *all'autoencoder*, ma che, partendo dall'estrazione di features dagli input audio e/o video, non restituisce un output della stessa natura, bensì la corrispondente ricostruzione dell'immagine di profondità, prodotta in valori metrici, in particolare in millimetri.

Partendo dalla credenza sulla cecità dei pipistrelli, la prima parte del lavoro è dedicata all'ottimizzazione dell'applicazione dell'ecolocalizzazione, scegliendo come punto di partenza la ricerca presentata nell'articolo [10]: la mappa di profondità delle stanze viene predetta direttamente dalle registrazioni audio. Questo passaggio permette da un lato la validazione del dataset acquisito, dall'altro la verifica della replicabilità di quanto introdotto dall'articolo, che è l'unico presente nello stato dell'arte che segue il filone delle applicazioni di machine learning per la predizione della profondità a partire dall'ecolocalizzazione e utilizzando dati reali.

Segue, quindi, l'analisi di reti neurali e parametri che possano ottimizzare l'estrazione di adeguati descrittori dai dati audio, per passare poi alle immagini a colori. Dopo aver prodotto alcune reti ed aver investigato lo stato dell'arte, i migliori encoder per questa applicazione sono stati riconosciuti in versioni personalizzate che prendono spunto da SoundNet otto strati [16] e VGG sedici [38] rispettivamente per audio e immagini.

Si passa in seguito all'integrazione delle due abilità di ecolocalizzazione e vista. Le architetture proposte sono varie (descritte nel Capitolo 3), ma le più performanti corrispondono alle due reti analizzate nel Paragrafo 3.7 e denominate *"Early Convergent Audio-Video Autoencoder Net"* e *"Two Independent Audio and Video Branches Net"*, le quali propongono rispettivamente convergenza anticipata e posticipata dei dati sensoriali.

Non limitarsi ad un'unica scelta è dovuto dal fatto che la prestazione di una prevale sull'altra in dipendenza dal tipo di applicazione e di dataset fornito per addestrare la rete neurale durante la fase di training. Difatti, la prima rete neurale opera la concatenazione degli embeddings dei rispettivi input audio e video, da cui l'immagine di profondità è predetta. Grazie a questa architettura (rappresentazione in Figura 35), la rete riesce a generalizzare più dell'altra e risulta migliore nel caso di dati di test che si discostano di molto dai dati forniti nella fase di training. La seconda rete (rappresentazione in Figura 42) presenta due rami che operano in maniera separata e restituiscono due diverse immagini di profondità: una derivante dall'input audio, l'altra dall'immagine a colori. Come ultima operazione, i due risultati sono forzati a convergere. Questo tipo di struttura permette di avvalere dei vantaggi dei singoli dati sensoriali e, come in alcune specie di pipistrelli, il risultato finale si affida al senso che fornisce una predizione più pregnante e, quindi, vicina alla base di verità. Il risultato è una rete che è in grado di imparare velocemente dai dati di training e che è valorizzata da scenari di test simili a quelli del training.

Per menzionare un possibile esempio, il primo caso eccelle rispetto al secondo nelle condizioni di training in un appartamento e di test in uno diverso; il viceversa si verifica nel caso di training e test nello stesso appartamento, se pur con un ridotto dataset nella fase di allenamento.

In ogni caso, l'introduzione di un secondo input di diversa natura rispetto al primo è motivo di miglioramento della performance, non solo dal punto di vista qualitativo (Figure 49 e 50) ma anche quantitativo (Tabella 10). Facendo riferimento alla mediana delle mediane calcolate sugli errori assoluti tra risultato e rispettiva base di verità durante il test ed applicando stessi parametri e dataset alle reti, l'applicazione dei soli dati di audio restituisce un errore di 431 mm, mentre la scelta della fusione degli input sensoriali porta nei due casi sopra menzionati fornisce degli errori di 375 mm e 426 mm rispettivamente (Tabella 11).

L'utilizzo di dati audio reali e nel campo dell'udibile porta con sé alcuni limiti; primo fra tutti la presenza di rumore, che non è completamente eliminabile, anche applicando un opportuno filtraggio. Inoltre, avendo scelto stanze di appartamento ammobiliate e con massime distanze misurabili ben al di sotto degli otto metri, non è possibile ottenere una registrazione del solo segnale riflesso, distinta da quella del suono emesso: le condizioni sono tali per cui si verificano fenomeni di riverbero e di sovrapposizione dei due segnali. In aggiunta, fenomeni di assorbimento del suono evitano del tutto l'acquisizione di alcune frequenze.

Nonostante le sfide ed i limiti sopra menzionati, la scelta di operare con il suono ha dei vantaggi rispetto ad alcune applicazioni nella visione, come il lidar, non trovando infatti difficoltà ad operare con finestre o specchi e non avendo problemi con superfici nere particolarmente assorbenti nei confronti della luce.

I contributi di questo lavoro sono innanzitutto legati all'applicazione di ecolocalizzazione per la predizione di profondità in stanze di normali dimensioni per un appartamento a partire da dati reali, seguendo un percorso diverso da ricerche nello stato dell'arte che si focalizzano su dati sintetici [9, 11] e discostandosi anche dal precedentemente menzionato articolo [10] che considera spazi ampi e con distanze massime di dodici metri. Inoltre, l'implementazione dei dati reali è stata applicata non solo alle reti neurali relative all'applicazione dell'ecolocalizzazione ma anche a quelle che integrano un approccio multimodale di fusione di immagini a colori ed audio. Un altro elemento da menzionare è lo sviluppo di un dataset condivisibile, validato per mezzo di una versione ricostruita e personalizzata dell'algoritmo di [10] e punto di partenza di un approfondimento di vari approcci al machine learning.

I risultati sono immagini apprezzabili da un punto di vista qualitativo, rappresentando le immagini di profondità in scala di grigio e in *colormap jet* usata dalla Kinect V2, come svolto anche in altri articoli; in aggiunta, le ricostruzioni sono espletate come matrici in valori metrici ed ogni pixel esprime la distanza assoluta tra la posizione dei sensori e quella di oggetti e pareti nella stanza, promuovendo il confronto dei risultati anche da un punto di vista quantitativo.

Testare le reti con un maggiore dataset è sicuramente uno dei possibili sviluppi futuri: nel gennaio 2023 gli autori di [10] pubblicheranno il loro dataset. Inoltre, per affrontare fenomeni di assorbimento del suono dovuti a specifici materiali, si potrebbe implementare una rete atta al riconoscimento del materiale a partire dal segnale acustico riflesso da esso. In aggiunta, si potrebbe approfondire la ricerca sul tipo di suono da emettere, facendo riferimento a quanto sviluppato dai pipistrelli o applicando lo sciocco della lingua usato da molti disabili non vedenti per l'ecolocalizzazione umana, a partire dal caso esemplare di Daniel Kish.

Codice e modelli sono disponibili presso *https://github.com/Elisa-code184/BatVision-and-depth-prediction-by-listening-and-seeing.git*

Index

1 Introduction

Humans perceive their surroundings using multiple sensory inputs including sound, sight, smell, and touch. Human perception is a process based on semantic information recognition from raw signals and extrapolation of cross-modal relations. The selection of the sensorial information relies on the task to accomplish: walking in the dark accentuates the auditory focus, for instance. However, [1] reports an interesting example: when we bite into an apple, not only we do taste it, but we also hear its crunch, see its red skin, and feel the coolness of its core.

Many animals, like bats and dolphins, use echolocation to estimate the distances of objects from them.

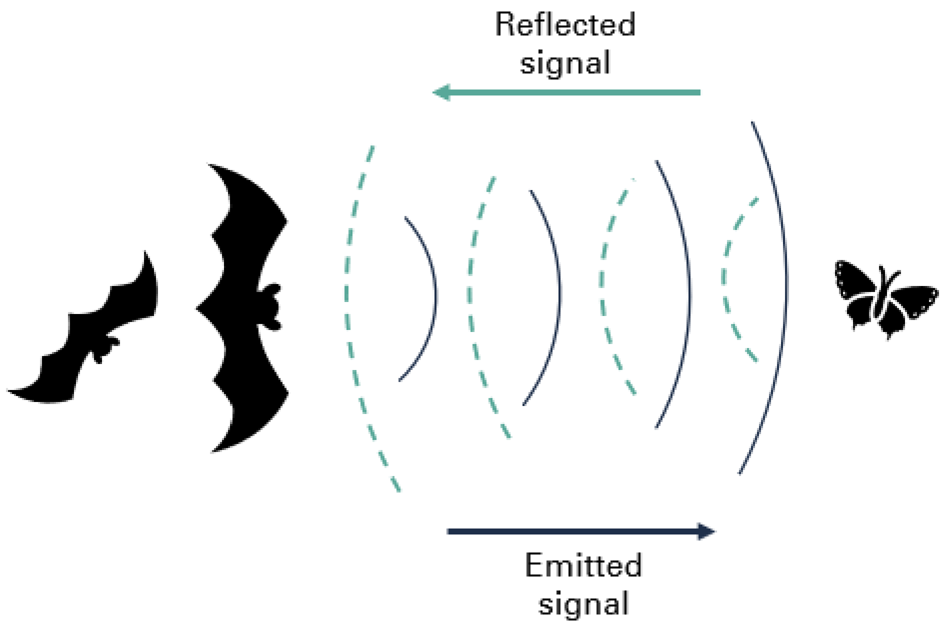


Figure 1. Simple representation of the echolocation found in bats

Echolocation is a process that is used to locate and identify a target by sending sound pulses and receiving the echoes reflected back from the target.

Bats can echolocate, meaning they receive echoes of self-produced sounds bouncing off objects to help them navigate. However, contrary to myth, bats aren't blind. Research shows that, depending on the circumstances and the species, bats sometimes prefer using eyesight to sound when hunting, some of them don't echolocate at all and have sharp vision, and some can even see ultraviolet light [48, 49].

Vision and echolocation are two modalities that allow distal sensing with high spatial resolution. The sensory integration is task dependent. For instance, echolocation allows bats to navigate in complete darkness. Bats integrate these sensory modalities, even when the presence of light would make it possible for them to rely solely on vision. Bats dynamically switch between the modalities: vision is given more weight when deciding where to fly, while echolocation is more dominant when approaching an obstacle.

Bats are ideal animal models to study multisensory integration; inspired by these animals, we address the problem of estimating depth with multi-modal audio-visual data, to infer the distance of objects with echolocation. The different approaches developed by analysing the bat multisensorial navigation can be broadly divided into three categories: (1) using audio modality only as the input, e.g. using echoes for depth prediction, (2) using visual modality as auxiliary information for an audio task, e.g. using videos to convert mono audio to binaural audio, and (3) using both audio-visual modalities together, e.g. for depth prediction.

Hence, the first approach followed in this thesis is described in (1), steering the problem of depth prediction to the audios in the audible range and acquired in real-world environments. Some recent methods have utilized echoes for depth estimation [9-11, 15-18, 25]. Echolocation does not present the same problems of vision systems, for instance represented by black light-absorbent surfaces, mirrors and glass for lidars. Furthermore, aerial acoustic communication is versatile because any audio interface-equipped device can be utilized as a communication device, i.e., hardware and operating system (OS) independent. Moreover, it provides an alternative communication interface for smart devices along with Wi-Fi and Bluetooth, which often are turned off to save battery and/or prevent unintended connections. Given the widespread use of smart devices that play, record, and process sound signals through their voice user interface, aerial acoustic communication under 22 kHz has recently been studied in depth.

However, camera applications are widely used [52]. The implementation of a webcam to get a monocular-grayscale image stream is simple and affordable. The fusion of audio data with videos makes the performance reliable, the results more robust against noises or critical conditions, and allows the handling of uncertainties, even without any notable need for higher costs. The neural network can adapt the performances and work in different conditions. Hence, the focus is consequently moved to (3), implementing the audio-video multi-modality, also to compare the possible advantages and disadvantages of using both audio and video, in contrast to only rely on echolocation.

The thesis is structured as follows:

- Chapter 1 – Introduction and state of art: presentation of goals and related challenges; short introduction to the main works related to the current activity.
- Chapter 2 – Materials and Methods: presentation of the approach, theoretical disquisitions, and explanation of the research steps.
- Chapter 3 – Experiments: presentation of the results, comments, and evaluation of the diverse machine learning applications.
- Chapter 4 – Conclusions: contributions and conclusions, future developments.

1.1 Research introduction

The research goal is obtaining depth maps in metric values through Artificial Neural Networks (ANNs) fed by recordings of audio reverberations and images, captured in indoor environments.

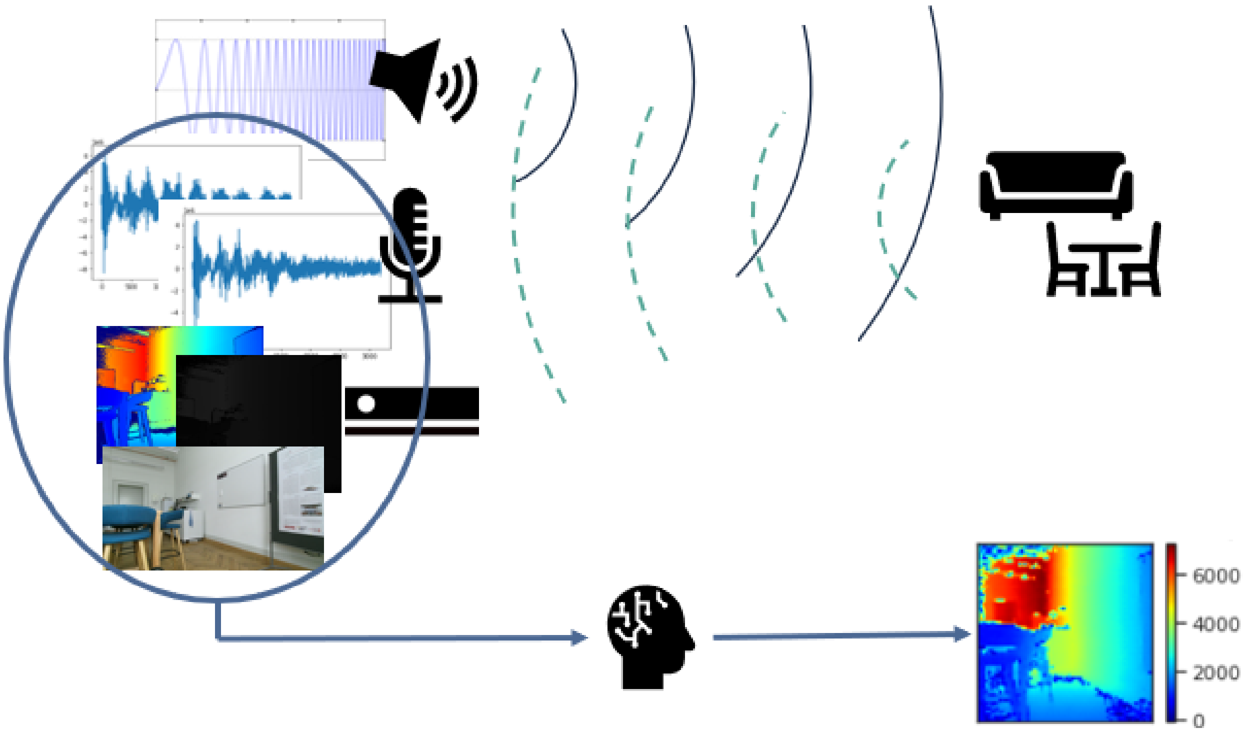


Figure 2. Simple representation of the research process

Specifically, the purpose is to demonstrate that it is possible to get reasonable predictions from machine learning applications, which take as input raw data, which are acquired in real-world indoor environments like furnished apartment rooms, utilizing affordable sensors and without the help of auricle-shape antenna to improve the capture of the sound in all the directions. The selection of ordinary sensors and the implementation of real-world raw data would like to match the possibility to contribute to future real-world applications, as previously done in [15-18, 25].

The contributions of this thesis are:

- collection of a sharable dataset of synchronized audio recording, RGB image, and depth samples in many diverse indoor environments;
- implementation of real data as inputs of the neural network;
- design and evaluation of multiple architectures and following assessments, evaluation and comparison between the depth predictions from audio modality and audio-visual modality;

- results delivered as depth maps in metric values.

Assumptions and downsides can be summarized in the following considerations. The data are collected in a quiet environment, specifically an apartment or meeting rooms, with different types of light conditions. Additionally, the sound signals of aerial acoustic communication are within the audible band. This has drawbacks: the sound signals heard by human beings can be disruptive, and the background noise distorts the acoustic signals. Furthermore, the room size limit is under the geometry that can allow an auditor to listen to the echo. In standard conditions, the distance needed to generate an audible and distinct echo by means of the emission of a sound is seventeen metres, considering the speed of sound in the air 343 m/s. The maximum size in any room is lower than eight metres; hence, the reflected sound overlaps with the emitted over time. Also, in small indoor environments with many objects and, consequently, various surface directions, such as furnished apartment rooms, the reverberation phenomenon is unavoidable: the emitted sound is reflected in many directions. Furthermore, the other related challenge is the phenomenon of audio absorption, which can be caused by many objects in a typical house: furniture elements with soft texture components can trap the audio waves and obstacle the reflection to the microphones.

Considering the challenges with audio data, the focus is moved to enhance the dataset quality, essential to have a functional training of a neural network; hence, in this thesis, the emphasis is in relation with the actual information extracted from raw inputs and the importance of the real-world data implementation.

1.2 Related works

Research into acoustic applications of machine learning has been divided into different branches, such as Sound Source Localization (SSL) and audio classification, also in the case of multimodal implementation. Sound source localization and audio classification have been proposed also in combination with video frame classification, with the purpose to find and classify sound-emitting objects in the scenes [2-7]. Additionally, the work presented in [8] focuses on audio classification: the authors achieved a large-scale and semantically rich representation of natural sound: they found out that semantics emerge in the sound network, even without supervised training.

These articles share multiple characteristics of their NN architectures. The audio is input as a time series [3, 4, 7, 8], or as a spectrogram [2, 5, 6]; however, the general NN model in many papers is composed of audio and visual sub-models for feature extraction that converge in some fusion layers [2-7]. Anyway, there are different approaches [8]. What is common is the use of convolutional neural networks to treat both modalities [3-8]; however, some exceptions are present, like in [2], which proposed a fully connected neural network. Regarding the image feature extraction, in many cases, the authors find a solution in popular encoders, also developed in response to challenges, such as VGG-Net [3, 5, 6], and ResNet-18 [4]. However, the research has also investigated different neural network options on that side [22-24].

The audio has also been involved in studies in the direction of echolocation, obstacle avoidance, depth prediction, and indoor environment representation. Echolocation can be described as the application of a biological sonar used to perceive spatial layout and locate objects in the world. In this field, some articles are more focused on the echoes, also in relation with visual navigation [9-11, 14]. The authors of [9] show that the echo implementation and learned image features help vision in spatial reasoning tasks, by presenting a novel interaction-based representation learning framework that learns useful visual features via echolocation. They simulate echoes in photo-realistic 3D indoor environments from the Replica dataset [13]. Replica is a dataset of 18 apartment, hotel, office, and room scenes with 3D meshes and high-definition range (HDR) textures, and reflectance information. Their echolocation simulation is based on a realistic acoustic simulation on top of Habitat [14], an open-source 3D simulator. Via simulation, they can isolate the echo recording from the original sound and noise, and they collected the synthetic samples every 0.5 meters in the 3D environment. In this case, the authors decided to provide the neural networks with audios as spectrograms.

The work presented in [10] proposes an end-to-end deep learning-based pipeline utilizing RGB images, binaural echoes, and estimated material properties of various objects within a scene. The authors used the material recognition dataset MINC for pretraining the material net, which is formed by the first four layers of the ResNet-18. The main part of the net is recovered from the article [9]. The authors of [10] tested their neural network on synthetic data taken from Replica [13] and Matterport3D [12], simulating the echoes through a precomputed RIR (room impulse response) provided by Habitat [14]. Similar to Replica, Matterport3D is a larger dataset containing 90 scenes. The room impulse response is the transfer function between the sound source and microphone. To recover the original sound source, the received microphone signal can be convolved with the inverse of the room impulse response function. Working with synthetic data allows to extract a clear pattern from the echoes in comparison with real-world data, since the audios present less noise and do not include the emitted sound.

The article [11] approaches the problem of depth prediction by emulating the echolocation mechanism. The prediction is only based on recorded audio, rather than relying on audio-video data fusion. The convolutional autoencoder takes as input audio time series containing both emitted and reflected sounds. The ground truth is stereo camera images.

The research has also approached real-world applications [15-18, 42]. The goal of [15] was to boost the accuracy of current systems based on standard optical cameras, through a real-time high frame rate generated acoustic map, overlaid over a standard optical image. The acoustic map is built up by a spatial-temporal filtering procedure, known as beamforming, that takes as input the signals acquired by an array of microphones. The use of acoustic imaging benefits the target tracking systems and consequently automated surveillance and security. In [16] the authors show a possible echolocation implementation, building a fully autonomous terrestrial robot that can move in the real world, avoiding obstacles with a system that applies the sonar principles. They train an image-to-image translation network to learn the mapping from recorded chirps and echoes from an environment to a 360◦ depth map of the environment. In [17], the authors proposed to combine a biomimetic navigation model which solves a simultaneous localization and mapping task with a biomimetic sonar mounted on a mobile robot to address two related questions. They adapted the mapping module of RatSLAM [31], a previously published rodent-inspired navigation system. The authors concluded that the biomimetic navigation model operating on the information from the biomimetic sonar allows an autonomous agent to map unmodified (office) environments efficiently and consistently. The work in [18] presents one of the attempts that have been made to build an actual robot that mimics bats' abilities, the 'Robat', a fully autonomous bat-like terrestrial robot that moves through a novel environment while mapping it solely based on ultrasonic sound. Using the echoes reflected from the environment, the Robat delineates the borders of objects it encounters, and classifies them using an artificial neural network, thus creating a rich map of its environment.

However, research has shown that echolocation is not limited to bats and dolphins. Visually impaired humans have also been reported to echolocate like bats, through mouth clicks or utilizing the long cane, and perceive the environment around them, like the presence of obstacles, perform distance estimation and object localization [19-21, 39]. The literature reports that between 20% and 30% of totally blind people may use echolocation; however, the research suggests that echolocation affords broad functional benefits. This is also the subject of real-case studies [40, 41]. In [40] it effectively provided the first report of psychophysical and clicking data taken from a group of 8 blind people with experience in mouth click-based echolocation performing the perception of distances through echolocation. In a recent study [41], the authors demonstrate that neither age nor blindness is a limiting factor in the human rate of learning or in the ability to apply echolocation skills to untrained tasks. This ability is trainable in humans, but not comparable, in terms of precision and versatility, to the highly specialized echolocation mechanisms found in bats and dolphins. The authors of [42] have focused their effort on helping impaired people with the support of a bat-echolocation system. The mouth click can be only in the audible range, but they have developed a device, which emits ultrasonic frequency-modulated sweeps. The echoes are recorded by bilateral ultrasonic microphones mounted in clay bat ears. This system allows human echolocation to be extended to the ultrasonic range, like in bats, and can open an interesting path in the machine learning application for echolocation.

The depth problem has been estimated also from monocular images [26-30]. In [26], the authors propose a convolutional neural network for image depth estimation with unsupervised training and a novel training loss, that enforces consistency between the disparities produced relative to both the left and right images. The paper [27] addresses the problem of estimating the depth map of a scene given a single RGB image: a fully convolutional architecture is there presented, including residual learning, to model the mapping between monocular images and depth maps. For optimization, they introduce the reverse Huber loss that is driven by the value distributions commonly present in depth maps. In [28], a self-supervised learning performs the monocular depth estimation. In [29], the authors address the problem to dense vision transformers. A transformer is a deep learning model that adopts the mechanism of self-attention, differentially weighting the significance of each part of the sequential input data, that are entirely processed at once. The vision transformer is used in [29] to get image-like representations at various resolutions and progressively combine them into full-resolution predictions using a convolutional decoder. The authors of [30] introduce a spacing-increasing discretization strategy to discretize depth and recast depth network learning as an ordinal regression problem.

2 Materials and methods

This study approaches the theme of depth prediction in an indoor environment investigating the potentiality of audio and audio-video cross-modal neural network potentialities for attempting to perform metric depth mapping in both cases.

Prior research [9, 11] already explored the possible use of synthetic data; in contrast, the data collection performed in this work is in real-world environments, like in [10]. At the moment, based on our knowledge of the state-of-art, a recognized real-world dataset of audios, monocular RGB images, and corresponding depth ground truth has not been published yet. Consequently, the first part of this work is mainly focused on, taking care of environment selection, data collection, sensor installation and setup, and acquisition algorithm implementation.

The collection of a dataset allows to move the focus on the machine learning investigation. The optimization of the audio-based depth prediction NN is firstly approached. The integration of a novel input stream to the network is secondly considered; in order to improve the accuracy of the depth estimation, multiple architectures, fed with audio and image streams to realize the multi-modality approach, are designed and tested.

2.1 Data collection

Starting from the first step of the workflow, this section introduces the devices used to collect data (Kinect, microphones, speaker) and the choice of the environments. Furthermore, here are discussed the issues that a relatively small indoor environment can cause with regard to the sound reflection and absorption (detailed explanation in *Appendix A*). Finally, the acquisition algorithm is explained along with all the related challenges.

2.1.1 Sensors and settings

2.1.1.1 Sensors

The sensors, needed to collect the necessary data for the NN training, have to be able to get audio, image, and depth image simultaneous streams and also to play audible sounds. Furthermore, since one of the aims of this work is investigating machine learning applications suitable for real-world tasks, to facilitate the possible future replication of the system in real-world applications, the sensors selected for the data acquisition are common devices.

The selected devices are two Sennheiser 66 microphones connected to the Audiobox USB for audio recording, the Kinect V2 for images and depth data, and the SoundLink Mini as the speaker.

Sennheiser 66 microphones are condenser microphones, which generally are more sensitive than dynamic and ribbon microphones. Condenser microphones are one of the most popular choices for recording applications because of their sensitivity and fidelity. In general, a condenser microphone provides a wider frequency response range than dynamic microphones, but a lower input sensitivity. In other words, they'll pick up more of the input signal faster. This means that the main part of the condenser microphones (with some exceptions) is better suited for quieter environments. These devices are essentially highly specialized capacitors. In a condenser microphone, one of the capacitor plates is made of very light, usually very thin material and acts as the diaphragm. The diaphragm vibrates when struck by sound waves, changing the distance between the two plates, and therefore changing the capacitance. The resulting fluctuation in capacitance creates an electrical representation of the acoustic energy from the input source. Active circuitry is required to transform the very high impedance of a DC-polarized capsule's output to a usable low impedance; +48V phantom power is commonly used to supply current to this circuitry. Other methods include dedicated power supplies (most often seen with tube condensers) and batteries (often seen with electret condensers).

The AudioBox USB is a mobile recording solution, with two front-panel combo mic/instrument inputs. This device is a mixer and permits the parallelization of the two audio stream data. Also, it allows connecting the two microphones through a single USB port and provides them with the right power.

The audio device datasheets set the limit of their application: the AudioBox USB working frequency range is from 20Hz to 20kHz; however, the microphone datasheet shows that the frequency range considered is 40Hz to 20kHz. Additionally, all the audio sensors are set up with a frequency rate of 44.1 kHz, reasonable value for the audible range from the Nyquist theory.

Separate microphones are needed in this work, because of the inability to access the single microphone audio stream of the Kinect array. This choice allows to control the audio acquisition synchronization between microphones, but also between microphones and the Kinect streams.

The Microsoft Kinect is a line of RGB-D sensors input devices produced by Microsoft and first released in 2010. Since its first appearance on the market, the Kinect sensor has been widely used by researchers because of its high potentiality, when used as a measuring instrument, combined with its very low cost.

The RGB-D sensors generally contain RGB cameras, and infrared projectors and detectors that map depth through either structured light or time of flight calculations, which can in turn be used to perform real-time gesture recognition and body skeletal detection, among other capabilities. They also contain microphones that can be used for speech recognition and voice control.

The Kinect V2 contains a Time-of-Flight (ToF) camera and determines the depth by measuring the time emitted light takes from the camera to the object and back. Therefore, it constantly emits infrared light with modulated waves and detects the shifted phase of the returning light. The accuracy in the determination of such distances is a function of the distance itself and may reach values of about 1.5 mm when the point is close (about 1 m) to the sensor.

Time-of-flight (TOF) systems are optical devices able to reconstruct 3D scene through the measurement of the time delay between the emission of a light ray and the reception of its reflection from a target. TOF cameras are TOF systems able to acquire a map of distances through diffusing light on the whole scene, perceiving the returning rays on a camera matrix sensor and measuring for each single pixel the phase shift between sending and returning light. Initially, TOF systems were built for the automotive market, particularly for car blind spot proximity measurements. Now, TOF systems are also implemented in other tasks. 3D depth cameras have had a major success with video games and entertainment, as mentioned before. However, their applications spread in different fields. Robotics is very rich of Kinect sensor applications, for instance in industrial cases, like no-contact Kinect-based methods for the safe cooperation between workers and robot manipulators, for monitoring industrial robots. Also, Kinect- based guidance and positioning of mobile robots or Kinect implementation as a component in autonomous ground vehicle for the indoor navigation.

A time-of-flight camera is composed of an infrared emitter, a matrix of IR sensors and an electronic circuit able to manage the signals and calculate the round-trip distance applying a proper mathematical model. The two most widespread TOF technologies are the pulsed and the continuous wave (CW). The first technique relies on measuring the time delay between sending and receiving a pulse with a fast counter synchronized with the emission. The distance (d) is carried out from the resulting delay (Δt) with the formula: $d = c\ \Delta t$, where c is the speed of light (in air) [36]. In the case of CW method, the emitted light is modulated with a sinusoidal or, more frequently, a square wave with frequency (f) in the range of 10–100 MHz. The CW method takes four samples per measurement, with each sample phase-stepped by 90 degrees. Using this technique, the phase angle between illumination and reflection, ϕ, and the distance, d, can be calculated: $d = c\ \phi\ /4\pi f$ [36]. The Kinect V2 utilizes the CW intensity modulation approach, which is most used in ToF cameras.

The Kinect V2, specifically, is composed by a RGB camera with resolution of 1920×1080 pixels, an infrared camera with resolution of 512×424 pixels and an infrared emitter. Also, a microphone array with four microphone capsules with each channel processing 16-bit audio at a sampling rate of 16 kHz. Microsoft provides a development tool (Microsoft SDK) useful to manage the Kinect V2 sensor.

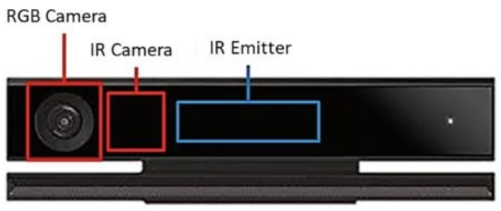


Figure 3. Kinect V2 sensors exemplification [51]

The data provided by the Kinect V2 are an RGB image with dimensions 1920×1080 pixels, an infrared image with dimensions 512×424 pixels, a depth image with dimensions 512×424 pixels and an array, with dimensions 512×424×3, in which the three channel values of a pixel in the IR image corresponds to the coordinates X, Y and Z of the spatial point in the camera reference frame. The depth image is effectively the infrared image, in which the grayscale value of each pixel has been replaced by the distance, in millimetres, from the sensor to the spatial point corresponding to that pixel.

The Kinect V2 specification for the depth is four metres and a half, a threshold disposed of concerning hand tracking. On the other hand, the Kinect V2 is able to sense until eight metres: this limit is set as the maximum depth perceived, and so, predicted by our neural network. The data considered in this work are RGB images and depth images on the Kinect side.

2.1.1.2 Settings

The recording rig is set on a trolley, whose height is seventy-four centimetres. This avoids the possible noises of sliding and allows to turn around and move the sensors with the minimum impact. The distance between the two microphones is set at thirty-four centimetres. The position of the sensors resembles what is shown in the article [10]. The Kinect V2 is positioned in between the two audio sensors and over the USB speaker. This configuration allows to play an audio signal by the speaker and record the reflections that bounced back from any obstacle in the indoor environment by means of the microphones, while the Kinect can capture the same scene. The two microphones resemble the two ears of a bat; however, no silicon auricle is applied, as presented in [10, 16, 42]. The sensors are set in a compact organization, an illustration of which is illustrated in the following figure.

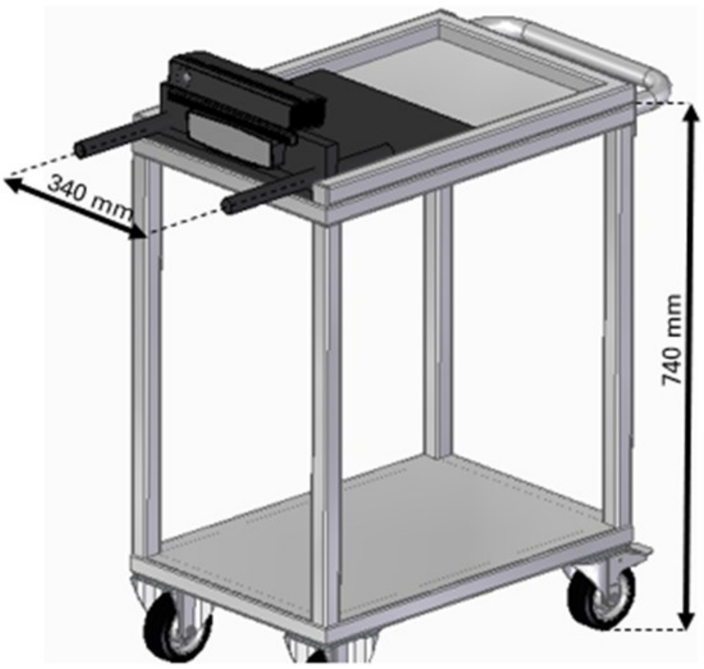


Figure 4. Sensor setup exemplification

The sensors, the Kinect V2 and the two microphones through the Audiobox USB, are connected through a USB connection in different ports to a personal computer. The speaker is connected though the headphone jack to the same device.

2.1.2 Environments

The environments are in the real world, indoor spaces in various light conditions. The variety of rooms means an assortment of different conditions of sound reflection and absorption, depending on materials, direction of the surfaces, humidity, temperature, etc. A brief of the sound absorption theory and a detailed description of the sound absorption property of multiple materials present in buildings can be found in *Appendix A.*

2.1.2.1 Scenario cases

In the state-of-art, the input data are mainly synthetic [9, 11]: in this case, the audio are simulated in a 3D reconstructed environment, like in Replica reconstructed apartments [13] and offices and the Matterport 3D reconstructed hotel rooms [12]. To pursue with the investigation of indoor environments, and to make feasible the possibility to realize real applications, the data in this work is acquired in an apartment and at the university, only considering comparable rooms dimensionally with the maximum measured dimension lower than eight metres, that corresponds to the Kinect system usage limit, contrarily to [10], where the authors mentioned that they recorded the audio data in wide spaces.

The issues, that can be caused by this choice, are related to the difficulty to recognize patterns of the reflected signals in the audio tracks. A time delay between the incident and the reflected waves, as explained in the theoretical extract (*Appendix A*), allows to distinguish the signals and is present only with distances higher than twelve metres in standard conditions. In real-world cases, the audios cannot avoid the presence of noise and reverberations. Moreover, only in some situations, for instance in empty spaces with flat walls twelve metres distant from the sensors, it is allowed to record the reflected sound separately from the emitted signal. However, furnished rooms of an apartments generally does not allow to do so. Hence, in the datasets collected in this work the time series signals recorded contain the emitted sound overlapping on reverberations and noises.

To clear the data as much as possible and to not make the environment unrealistic, the rooms are cleaned up from unnecessary objects but kept as natural as possible. The purpose is keeping the reverberation energy high. From the theoretical research, it is clear that the needed surfaces are the ones that can reflect the sound, in other words, their sound-absorption coefficient should be low as much as possible. The estimation can be conducted taking into account the following considerations: the surfaces need to be hard, dense, orthogonal to the sound direction and plate. With opposite characteristics, the effect is to trap the sound, and the material results to be soft and porous (*Appendix A*).

All the mentioned annotations guide the selection of the environments to collect data suitable for the task of this work. Many trials are done in order to acquire data in different conditions. The datasets produced are mainly two: *Dataset 1* is composed by 800 coupled elements (RGB image, depth image and two audios), *Dataset 2* is 4975 groups of coupled data. *Dataset 1* is gotten in an apartment of six rooms and a hallway in quiet conditions. The process starts with the installation of all the systems and the trolley positioning in a fixed point of the room, then it proceeds with turning the trolley around its position and moving it back and forth. However, the differentiation of the scenarios is mainly provided by moving the object around the sensor setup. *Dataset 2* is acquired for the main part in the same apartment and 1281 data groups are collected at the university. In this case, the acquisition process starts with the installation of all the systems, as well. However, the different scenarios are collected moving the trolley to cover all the possible directions of a room in order to have not only diverse objects and their positions, but also to differentiate the architecture that characterizes the portion of the room captured. This allows to differentiate more the dataset, that helps the training of the neural networks, specially to recognize sound patterns and their relation with the room geometry. Moreover, the rooms chosen from the large range of possibilities available at the university are small meeting rooms, also considering their private kitchens. The selection of the room is done considering a maximum measurable distance of eight metres, that is the distance limit written in the datasheet of the Kinect V2.

Fixing the position of the setup in every room, a dataset of 70 paired depth images, RGB images and audio recordings is acquired for each different scene. The points of the room selected as significant sensor setup positions are especially the corners that allows to cover a wide portion of the room, and differentiate architecture, furniture and objects turning around the sensor setup. In the following figures are examples of RGB images and depth images (in grayscale and jet colour map) acquired.

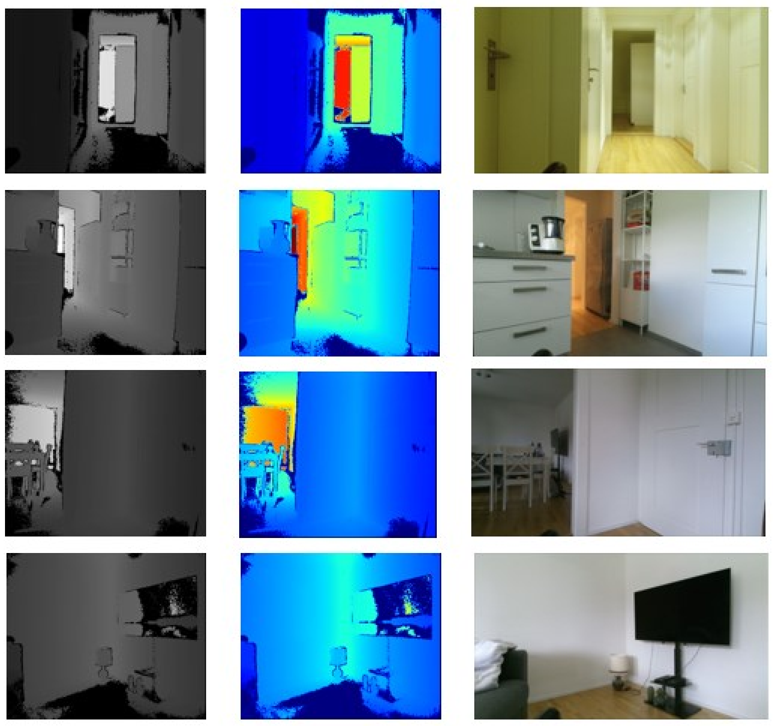


Figure 5. Examples of the Kinect samples acquired in the apartment. In order, grayscale depth image, depth image in jet colormap, RGB image of an apartment scenario

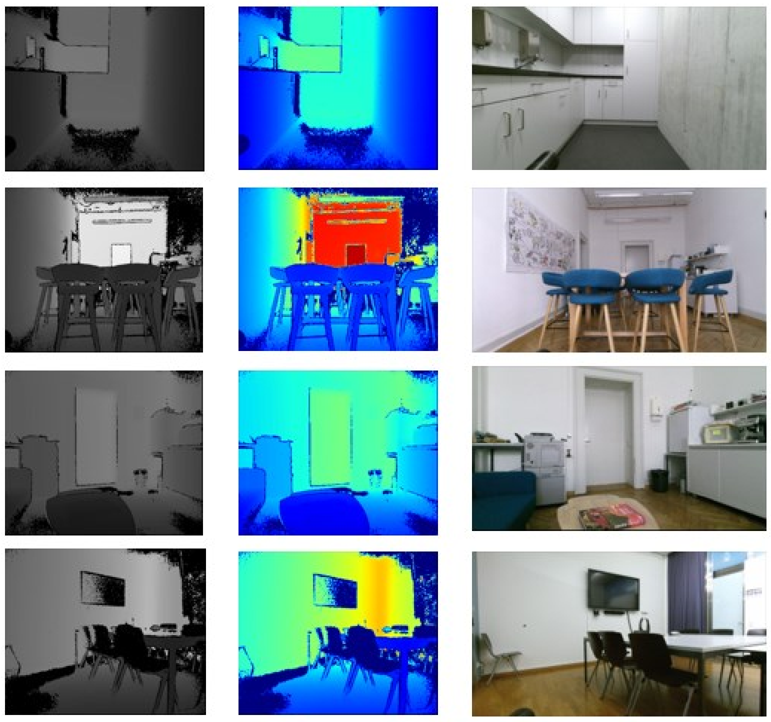


Figure 6. Examples of the Kinect samples acquired at the university. In order, grayscale depth image, depth image in jet colormap, RGB image of an apartment scenario

2.1.3 Sound selection

In this section the selection of the sound which is played by the speaker is introduced. This sound represents the emitted signal, which realizes the echolocation and should provide the suitable reflections for this task. Examples present in nature are bat and dolphin echolocations and emitted sound, investigated and applied in [17, 18]. The concept comes also from human echolocation, in which the use of mouth clicks is documented in [19-24]. The clicks took the form of highly focused sound waves emitted in a 60-degree cone, compared with 120 to 180 degrees for typical speech. The echolocators send their clicks in many directions toward the space they were sensing [50]. Ranging in frequency from 2 to 4 kilohertz, the clicks are higher in pitch than speech, perhaps because it helps keep the cone of sound tightly focused. The clicks are also very brief, lasting only 3 milliseconds, which might help avoid the emitted sound overlapping with its echoes.

However, many articles document the use of a frequency modulated signal called *chirp* [9-11]. Furthermore, the research, presented in articles [9-11, 16, 17], introduce the use of a sound which covers the audible range, in order to provide reflections in the audible range as well [9-11, 16, 17]. As in human echolocation estimations, the time duration of the track is set at three milliseconds, and the chirp frequency increase linearly to involve all the audible frequency range.

The ideal frequency range, for what mentioned before, is from 20 to 20k Hz; however, the datasheet of the selected microphone shows that the maximum range the sensor can perceive is from 40 to 20k Hz. Therefore, that is the choice for the sound frequencies. The selected sound to be played as the emitted signal is a chirp. It increases its frequency from a minimum to a maximum, in this case it covers all the frequencies from 40Hz to 20kHz. The selection of the chirp is done to make results and performances of this research comparable to the state-of-art [9-11], but also to cover all the frequency range and shape the sequence of the frequencies in an organized and recognizable short sound.

Its duration is fixed to 3 milliseconds, because of the need of an extremely short audio: the emitted sound is going to overlap over time with the reflected signals. Since the audio data are provided as time series to the networks and the audio pattern to be recognized is on the amplitude samples, the superposition of the emitted over the reflected signals darken the significant information for this work. Considering a sound velocity of 343 m/s and that the mentioned maximum distance is eight metres, sixteen metres are necessary to let the incident wave to reflect; the ToF of the sound is 0,04664723 seconds. Even for the maximum distance in the dataset, a chirp of three milliseconds overlaps for the 6,43%.

The chirp is initially produced by means of the library SciPy; however, the Audacity synthetic chirp demonstrates a higher quality, the sound is less noisy: the choice is producing a track in .wav format and directly feeding the code. Audacity is multi-track audio editor and recorder for Windows, Mac OS X, GNU/Linux, and other operating systems; it is also mentioned and suggested in other articles [9, 10].

The settings to generate the chirp track in Audacity are the frequency range and the time duration, but also the type of interpolation and of waveform, respectively set up as linear and sine. The first idea was creating a track of three milliseconds, however, because of limits explicated in the following paragraph the chirp is followed by half a second of *silence*, option available in Audacity and corresponding to a time series of zero amplitude. In this way, every chirp is distinctly separated from the next one by silence, in which the speaker is not turned off or in stand-by, but still plays a track of zero amplitude. The track can be exported as .wav file. The chirp in the time domain can be visualized below.

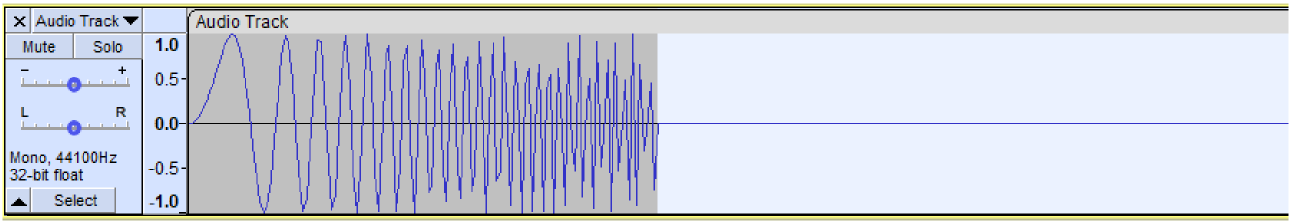


Figure 7. Chirp signal in time domain represented in Audacity

2.1.4 Acquisition algorithm

This section is dedicated to firstly introduce the investigation of the libraries that allow the synchronization of all the actions needed for the data acquisition, and secondly explain the algorithm in detail.

2.1.4.1 Libraries and tools

The initial investigation was conducted with the *PreSonus* application of *Studio One*. Studio One is a digital audio workstation application, used to create, record, and mix audio. Initially developed as a successor to the KRISTAL Audio Engine, it was acquired by PreSonus and first released in 2009 for macOS and Microsoft Windows. The need of synchronization between the two microphones and between playing and recording is satisfied by Studio One. However, the Kinect V2 synchronization and frame acquisition was an issue.

Hence, the data collection is handled by a Python script supported by a different library for audio and video. For the Kinect side, the libraries investigated are *PyKinect2, libfreenect2, freenect 2.0, and ktb*. libfreenect2 is the driver for Kinect V2 devices: it comprehends a set of helper functions to make using the sensor with Python. The available applications are RGB image transfer, IR and depth image transfer, registration of RGB and depth images, a robust interface. The freenect2 module provides a Python interface to the libfreenect2 library. Another investigation was with the library ktb*, kinect-toolbox*: with this library the Kinect V2 can be used more like a cv2 webcam; this package provides methods to get colour, depth, registered colour, registered depth, and IR images, record video, get point clouds. PyKinect2 is one of the Kinect projects that support the Kinect V2 stream control through Python scripts. It enables writing Kinect V2 applications, games, and experiences using Python.

Since the use of the Kinect V2 is only related to the need of capturing the stream of depth and RGB frames, all the mentioned libraries are suitable to accomplish the task. The motivations that drive to a deeper investigation are connected to the need of synchronizing the image collection to the chirp playing and audio recording. The library, which allows to get access to single frames and can work with the other selected libraries for audio and running parallel processes, is PyKinect2. Moreover, it provides different examples of possible implementations, is easily approachable and saves data rapidly. The dependencies are Anaconda 32-bit version, Kinect for Windows SDK V2 and the library *comtypes*. After the first issues in satisfying all the requirements and in matching the libraries versions (https://github.com/Kinect/PyKinect2/issues, issue #102), the library supports the user in getting the RGB image stream and the raw depth stream, respectively in the system types of uint8 and uint16.

On the audio side, the investigation focused on *PyAudio, windsound, Pydub, Sounddevice, Simpleaudio*, among others. PyAudio provides Python bindings for PortAudio, the cross-platform audio I/O library. PyAudio allows to play and record audio by means of Python on a variety of platforms, such as Linux, Windows, and macOS. The winsound module is a Python library customized for Windows audio-playing platforms. Pydub is a very powerful library, however, there is a number of dependencies. Like in PyAudio, it can directly read .wav file, however it needs ffmpeg or libav to access .mp3 audio. Sounddevice provides bindings for the PortAudio library, as PyAudio, and functions to play and record NumPy arrays. Simpleaudio supports audio playback in Python, and it has no dependency.

The selected library for audio recording and playing is PyAudio. PyAudio is versatile on Windows operating system and outperforms in this application, in which playing and acquiring audio recordings at the same time is a necessity for the proper dataset collection. Also, it is the only library that allows to synchronize audio recording and playing with the kinect acquisition through the application of the multiprocessing library, described below.

The synchronization of image and audio stream acquisition is ensured by the multiprocessing library for Python. The first trials were supported by the threading library, however the attention was moved to the multiprocessing library because of the following reasons.

By formal definition, multithreading refers to the ability of a processor to execute multiple threads concurrently, where each thread runs a process. However, multiprocessing refers to the ability of a system to run multiple processors concurrently, where each processor can run one or more threads. Multiprocessing allows to create programs that can run concurrently (bypassing the GIL) and use the entirety of your CPU core. The Python Global Interpreter Lock, or GIL, is a mutex that prevents multiple threads from executing Python bytecodes at once, even in a multi-threaded architecture with more than one CPU core.

2.1.4.2 Code description

The algorithm is organized in three functions, one per each sensor acquisition or playing, that run at the same time by means of multiprocessing. The data collections are independently managed by the different sensors, in a synchronized manner thanks to the multiprocessing implementation.

The main function of the script is dedicated to the call of the multiprocessing execution, as the multiprocessing library requests. In the main each function is assigned to a *process*, then the processes are stored in a list. The multiprocessing execution is sequentially triggered for each element of the list: *process.start()*. By adding the *process.join()* for each process, the interpreter does not proceeds with the execution until all of them are completely closed.

The three functions call the sensors classes: Microphones, Speaker, and Kinect, developed in a second script. The structure is almost the same for each class: initialization, play, close and save.

The speaker class is initialized by setting all the parameters and variables: define file name and the length of chuck (1024), upload the file with *wave.open*(), create an interface to PortAudio by *pyaudio.PyAudio(),* initialize the stream by *pyaudio.PyAudio.open()* declaring format, channels, rate. PortAudio [37] is part of the PortMedia project, which aims to provide a set of platform-independent libraries for music software. PortAudio is an open-source portable audio I/O library designed for cross-platform support of audio, for instance for audio playback and recording. It can run on many different computers operating systems, including Windows, Mac OS X and Linux. The next methods are *self.play()* and *self.close()*: the first one allows to play the entire file, read the entire stream by *wf.readframes()* and rewrite it chunk by chunk to read each frame at time; the second one lets close the stream and terminate the PortAudio interface.

The Microphones class has a closed configuration to the one of the Speaker. Firstly, the initialization defines all the stream parameters (recording duration, chunk length, sample format, number of channels, sample rate) and triggers the interface to PortAudio. After establishing a new frame list and initializing the stream, the stream is open, through the interface by pyaudio.PyAudio.open(). The record method allows to store data in chunks for *n* seconds, instead the close method closes the stream and terminate the PortAudio interface. Open a new file in which write the frames following all the set parameters are the functionalities of the *self.save()*.

The Kinect class can be initialized by means of PyKinectRuntime.PyKinectRuntime (PyKinectV2.FrameSourceTypes_Depth, PyKinectV2.FrameSourceTypes_Color), that allows to start the runtime. This is followed by the extraction of the frame dimensions for depth and RGB image by means of the PyKinect2 get methods. *self.play()*, a method that permits to obtain the Kinect V2 data stream, must be in a while loop; a OpenCV wait key is set to eventually break the loop. In this particular application, the wait key is 30, that corresponds to the Esc button. The play routine allows to get the depth and RGB frame, reshape with the same system format, apply the OpenCV JET colormap and save the images. These functions are accessed if the Kinect methods related to check the availability of a new depth and RGB frames corresponds to True.

The Kinect V2 system format is *uint16* for the depth image and *uint8* for the RGB image, hence the format used to save the frames is *.png* by means of the function *imwrite()* of OpenCV that is a lossless format for both *uint8* and *uint16* data. The choice to save the frames not as a matrix, for instance a Numpy array, came from the need to monitor the data during the acquisition. The JET colormap from the library OpenCV is the standard for the Kinect V2 depth output. The last method, *.close()*, has the purpose to end the Kinect stream and OpenCV call.

The first version of the script applied the multiprocessing library to acquire single audio tracks by the microphones while playing single chirps through the speaker and getting single images from the Kinect V2.

The purpose was having group of associated cross-modal data in loop. To get the series of acquisition automatically, the main function was set in a while loop to ensure the continuity.

It is possible to get single tracks of the desired dimension, although the synchronization is lost after a few acquisitions: the reproduction of the sound is perfectly functioning, however, in the saved data, the chirp is not acquired completely or cut, or in the worst case, not included. The issues are in the different lengths of the processes, in the computational power of the computer (four cores), and the main problem should be found in the position of the multiprocessing activation: it is included in a while loop.

It is possible to smoothen the time differences in the execution of the multiple functions by introducing an enforced time delay in the faster routine. The introduction of *time.sleep()* is a commonly approached in diverse routine codes, however this solution makes the timing is uselessly long: from two to four seconds to obtain just one data group (audio, image, and depth).

Since the multiprocessing is meant to be used in the main function of a script, being included in a while loop caused issues in the parallelization of the processes. Even applying the *.join()* method to force the loop to restart after having closed each process, the issue is not solved. The processes are not all completed before the loop restarted from the first one. It also can depend on the shortness of the chirp track intended to be played, three milliseconds, and of the recording, half a second. Another reason relies on the low number of cores of the personal computer.

The answer to the problem was found in moving the while loop inside the single processes and leaving the multiprocessing as the main function. To do so, the chirp reproduction, particularly *Speaker.play()*, is set in a while loop and the recording is extended to include many tracks. The Kinect acquisition is included in a while loop as well, as default. The timing is now set to one second for each data group.

This approach shortens the acquisition time and solves the synchronization issues; however, the data pre-processing routine has to include a customized process for the isolation of the single audio samples from the continuous recording. It is possible to recognize that in [10] algorithm the authors decided to randomly cut the needed number of samples from the audio stream. It might mean they handled the issue in the same way.

2.2 Data pre-processing

Data processing is a necessary step to prepare the samples in order to train the artificial neural networks. At the beginning of this process, the data are represented as a long audio stream comprising of both the audio channels, RGB images with a resolution of 1920×1080 pixels, depth images with dimensions 512×424 pixels. The samples of the different modalities must be grouped together. The function of the data pre-processing is to create a series of groups of synchronized left and right channel audio recordings, RGB image and depth image. The processing involves audio channel division from the recording, resolution downscaling for the images and audio and image coupling and saving; a detailed description follows.

The first step in the data pre-processing is uploading a sorted list of the image paths and the audio path. The array audio stream is uploaded: it corresponds to the amplitude time series that contains chirps and relative reverberations. The channel division is applied to obtain two distinct arrays, before applying the bandpass filter to each single microphone track.

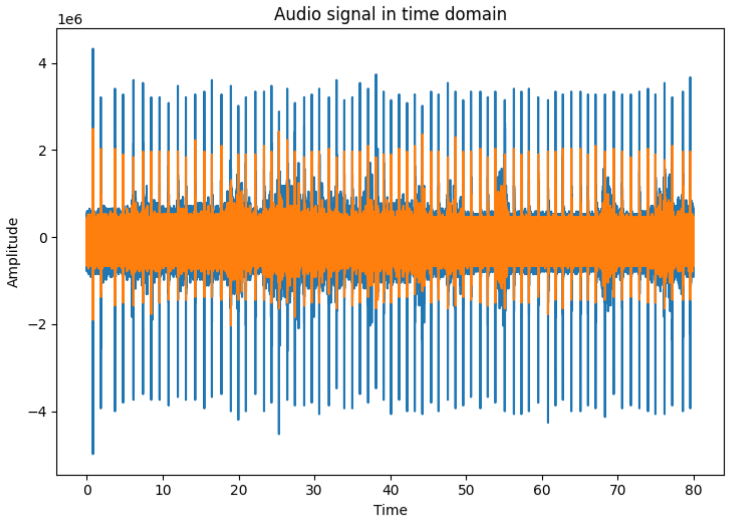


Figure 8. Superposed channel time series

The bandpass filtering relies on the Butterworth filter. The Butterworth filter is a type of signal processing filter with as flat as possible frequency response in the passband. The bandpass filter is needed in this application in response to the selection of the audible range as the sound range for the audio recordings. The purpose is to get rid of the portion of the acquired signal that corresponds to frequencies different from the audible range.

This work applies the *Scipy.signal* version of Butterworth digital and analogue filter, which is presented with the low and high frequencies of the band. This variable, united with the second order and the "bandpass" type, is fed in the *Scipy.signal.butter()* function. The obtained filter coefficients are the input of *Scipy.signal.filtfilt(),* together with the audio signal. This function is the actual element that applies the filter to the signal and allows one to get the filtered audio.

The audio samples, containing emitted and reflected sounds, are collected in a longer audio recording, and need to be divided in single data. A simple operation of cutting the stream every half a second is not an accurate method for this application and can affect the data effectiveness and synchronization. The thresholding is made necessary because, despite the multiprocessing enforces the process synchronization, it does not control the timing needed to complete all the functions during a loop. The functions can be delayed by fractions of seconds; however, it is enough to be tangible in a cutting operation and affecting the result accuracy and synchronization.

Filtering by *Numpy.where()* allows getting a tuple of the indexes in correspondence to the elements that fulfil the condition indicated as the argument; in this case, the argument relates to the thresholding condition, and it is satisfied by the indexes of the elements in the audio array that are higher than the numerical threshold. It can be applied to both microphones; however, in this work, only a channel recording sees the application of the thresholding condition, and the resulting indexing is used as a reference for the other channel.

Since the acquisition of a data group and the chirp playing happens every second, the purpose is to get an audio sample at the same timing. Hence, the considered audio values are in correspondence to the indexes in the indexing tuple that are at the minimum distance of 42000 samples from the previous one.

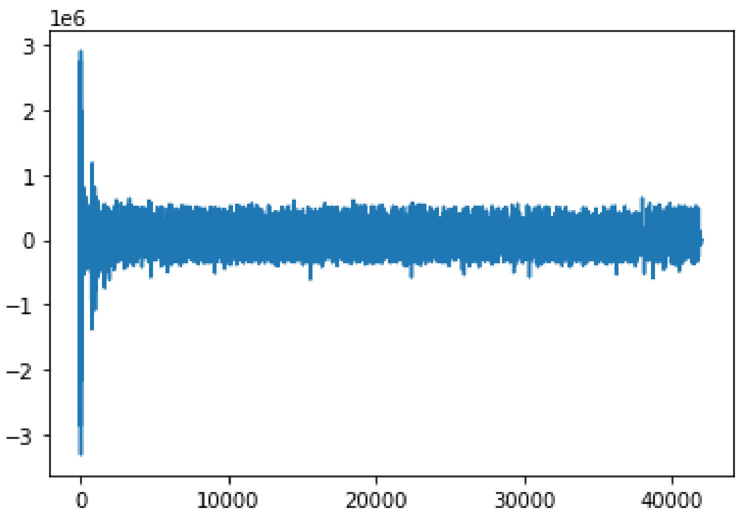


Figure 9. An audio example cut from the long audio recording and resulting after the thresholding application

Each output track is in time domain and composed by 42000 amplitude elements, hence, shorter than one second, and containing chirp and relative reverberations.

The track is cropped 200 samples before the thresholding index and 41800 samples after, to smoothen eventual indexing errors. The second crop is accomplished by taking into consideration a random element between 0 and 200, as a starting key to get 2200 samples, as done in [10] code. However, the last applications see a fixed starting key corresponding to zero.

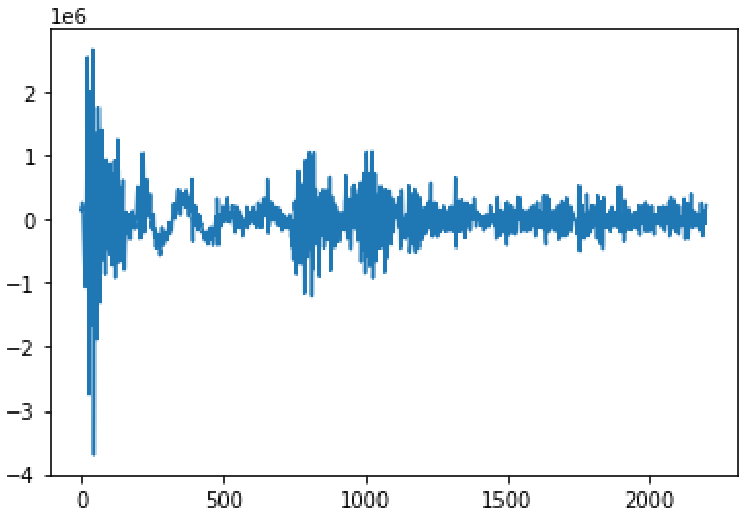


Figure 10. An audio example resulting from the second cut

The choice of getting 2200 samples is the result of the following considerations: the audio must comprehend a chirp of 3 milliseconds and all the reverberations for a maximum distance of 8 meters, considering that they end with the reverberation of the last instant of the chirp. With a sampling rate of 44100 Hz, the sum of 0.003 and around 0.046647 seconds corresponds to 2200 audio samples.

Since the second crop is guided by a random key, it is effectuated in the *DataLoader*, the data primitive available in Pytorch and that allows to decouple the dataset code from the model training code in order to improve the modularity. The successive audio processing operations, i. e. normalization, and reshaping of the array from (2200) to (1, 2200, 1), are completed in the DataLoader function.

The audio-image associations and the image operations are included in the pre-processing function. Anticipating these procedures validates storing processed data in an H5 file; the data are ready to be inputted into the artificial neural network. It avoids the waste of computing power and time at every new training.

The h5py package is a Python library for the HDF5 binary data format. It lets store huge amounts of numerical data, and easily manipulate that data from NumPy. H5py uses straightforward NumPy and Python metaphors, like dictionary and NumPy array syntax. Thousands of datasets can be stored in a single file, categorized, and tagged in customizable ways. This is the power of H5 files, but also a downside when the purpose is the sharing of the datasets. In fact, the access to the h5 requires to know the structure, the labels under which the needed data are stored. The h5py files are in a widely used standard binary format, which you can exchange with other people, including those who use programs like MATLAB. The opportunity to store data of different types and lengths is favourited by the selection of NumPy to organize the audio signals and images.

The data, saved during the acquisition process, are named with the machine time; hence any file name is associated with the number of seconds elapsed from the 1st of January 1970 to the very last instant at which the acquisition happens. Moreover, the RGB and depth image collected at the same Kinect acquisition loop is associated with the same machine timing.

The name of the images also differs from type to type, which permits the sorting of the image paths in different type lists. From the sorted list of the image paths, uploaded at the beginning of the pre-processing function through *glob.glob()*, the RGB image paths are divided into the depth images and depth colormap images. Therefore, the paths are cropped to isolate the machine time, present in the file name, and from which the number of seconds of the first audio track, obtained from the previous investigation, is subtracted.

The purpose is to have a common starting point for audio recordings and images that enable to make the comparison and to evaluate the time delay between audio and image samples, considering their relative time delay from the beginning of the acquisition. Consequently, dividing all the audio sample indexes, stored in the resulting list from the thresholding process, over the sampling rate, the resulting values are the time intervals elapsed from the beginning of the data collection to the acquisition of the single samples.

The comparison between the timing of audio recordings and images is executed in sequence, starting from the first audio and the first image. The accepted error is half of a second, which can be accepted as a reasonable amount, considering that the time between two consecutive acquisitions is one second. The relative time delay is calculated as the absolute value of the difference between audio and image time. If the error is major than half a second, the sample with the lower delay is deleted, because the other one surely matches its time delay with the next sample of the other modal series. The comparison restarts from the beginning when any error is found. An additional evaluation is the comparison of the number of elements present in the resulting audio and image lists. Hence, the number of samples is enforced to be equal.

The operations related to the images are applied only to the ones that are selected as part of the actual dataset going to feed the ANNs. The first step is the upload by means of *OpenCV.imread()*: regarding the

RGBs, the three channels are uploaded, then switched from BGR (OpenCV channel standard) to RGB, since the standard order for OpenCV is BGR and the image channels are read swapped compared to the other more usual standard RGB.

The use of Colour images avoids the risk of failures in image feature extraction and consequently associations between features and depth reconstructions, especially in a strange correlation between grayscale images and pixel intensity values to the depth metric values.

Concerning the depth images, having chosen to work with the depth raw data from the Kinect, grayscale in uint16 type, the upload is executed by means of the *OpenCV.ANYDEPTH()* option. Both image categories are involved in Numpy.float64() conversion and downsampling. Then, RGBs are subjected to reshaping from Height x Width x Channel to Channel x Height x Width, differently from the depth arrays, to which a dimension is added at the first position.

2.2.1 Image downscaling investigation

The selection of the image resolution to achieve during the downscaling is fixed during the first training of the first multi-modality network. The first experiment consists of consigning RGB images with dimensions 128x128, the depth images resulting from the net are defined with the same shape.

However, the memory request was too high for a natural prosecution of the work. In many trials, it delivers execution issues working with Colab Pro Plus. The time request for Euler service to accept the script execution is too elevated.

Hence, the effort is focused on decreasing the request of memory in relation to the execution of the neural network: the reductions are applied in the number of layers, kernel size, etc. Also, the data are provided through an h5py file. After many trials, the attention is moved to the image resolution. Through the following training executions, the net shows it is able to complete the training of one hundred epochs successfully.

The first trial is downscaling the image dimension to 28x28. However, the following labels are not enough sharp to conduct a depth mapping evaluation.

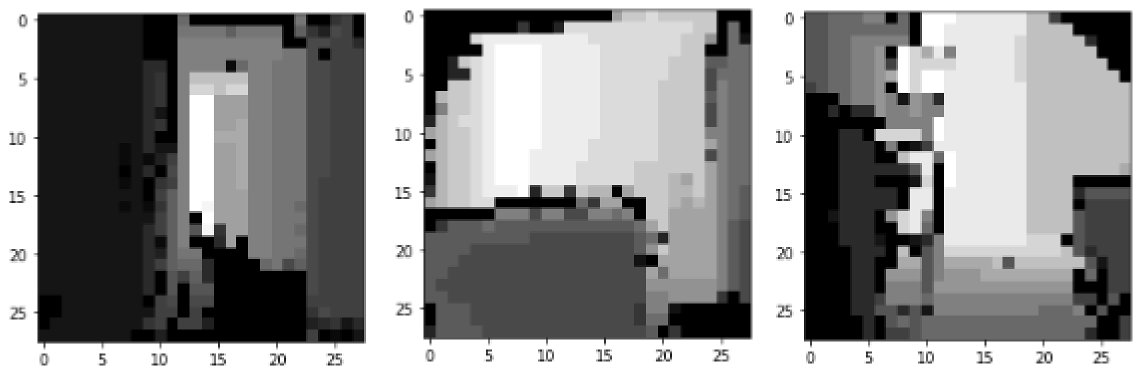


Figure 11. Examples of grayscale images, downscaled to 28x28 and then taken as input in the network

Hence, the resolution finally considered is 64x64, which is also the selection of other popular networks [38].

2.3 Artificial Neural Networks

The problem of obtaining the depth maps from audio reverberations and image acquisition is assigned to a regression approach. Regression algorithms are supervised learning algorithms, implemented in Machine learning and that work with the labelled datasets. Regression algorithms are used to predict the continuous/real values, such as depth values in this work. Regression analysis is a statistical method to model the relationship between dependent (target) and independent (predictor) variables with one or more independent variables. More specifically, regression analysis helps to understand how the value of the dependent variable is changing corresponding to an independent variable when other independent variables are held fixed.

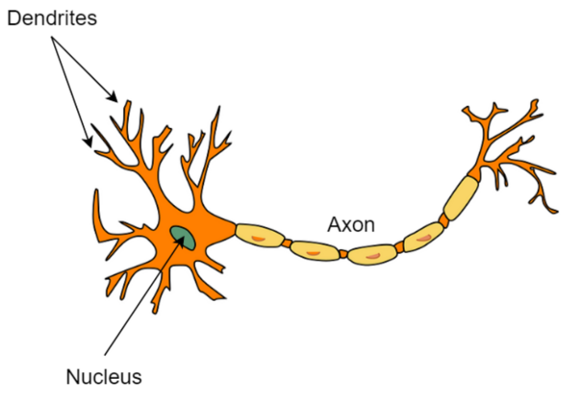


Figure 12. Representation of a biological and an artificial neuron [53]

A NN is based on a collection of connected nodes, called *artificial neurons*, which are the interpretation of the neurons in a brain. Each connection, metaphorically the synapse, can transmit a signal from a neuron to another by means of the *edges*. The artificial neuron, that receives the signal, processes it and transmits it to the connected neurons. The shared message is a real number, and the process, that each neuron applies, is a non-linear function implemented to the sum of its inputs.

Neurons and edges typically have a weight that adjusts as the learning proceeds. The weight increases or decreases the strength of the message. The neurons are organized in layers, the signals are processed at each layer and then sent to the next one. The neurons of the same layer apply the same function. Neural networks can be defined as a set of dependent non-linear functions.

The neuron mathematical model can be described by the following figure and formula, where the *x* and *y* correspond to inputs and output, the *w* are the weights applied to all the inputs and *b* is the bias; a linear function is applied to these values and followed by an activation function *f*.

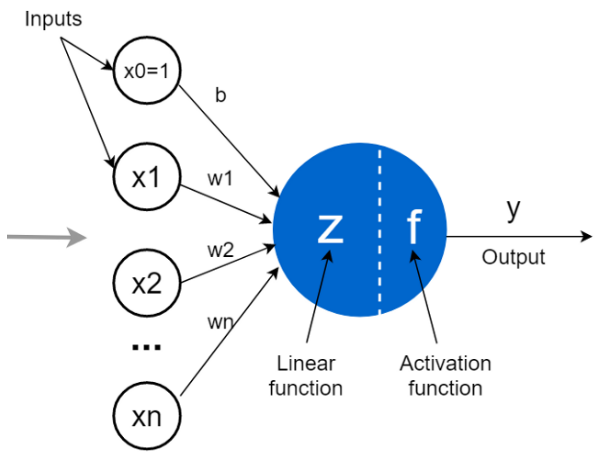


Figure 13. Mathematical model of a neuron [53]

$$y = f\left(\sum_{i=1}^{N} w_i * x_i + b\right)$$

All the proposed ANNs are trained as supervised algorithms; however, they are meant to be utilized during the test in unsupervised applications. The nets should be able to predict the depth from the audio and video inputs applying at each neuron the parameters learned during the training phase and stored in a pre-trained model.

The training process allows an ANN to learn: the ANN modifies the weights of the connections during this phase. A way of learning is known as *back propagation* and defined as a supervised learning process. In a supervised training, the ANN calculates the error between the output and the original data, or “label”, and determines how far its guess was from the actual answer. The error is obtained applying a *loss* criterion and utilized to adjust the connection weights in order to reduce the gap. The optimizer is an algorithm that modifies the attributes of the neural network, such as weights and learning rate. It helps in reducing the overall loss and improving the accuracy. The learning rate is a hyperparameter that controls how much to change the model in response to the estimated error each time the model weights are updated. More details about loss criterion and optimizer are presented in the next paragraphs.

Neural networks are widely implemented because of their power: they can recognize patterns, extrapolate information, reproduce synthetic data that emulate reality, etc. In this work, multiple neural networks are investigated in regards with audio and image feature extraction and depth map prediction. The common thread is an NN architecture described in the next paragraph.

2.3.1 Autoencoder

Autoencoder is a type of artificial neural network used to learn compressed representations of data. It was initially developed for machine translation problems or, in other words, sequence-to-sequence prediction problems.

The architecture consists of two sub-networks: encoder and decoder. The encoder compresses the input in an embedding, also called *encoding*: it contains the descriptors and is in the latent space, which corresponds to the bottleneck between the two sub-models. The decoder predicts an output, given the intermediate vector; the decoder attempts to recreate the input from the compressed version provided by the encoder. The two sub-networks usually have related architectures: the decoder layers reverses the encoder layers. The overall architecture is shown below.

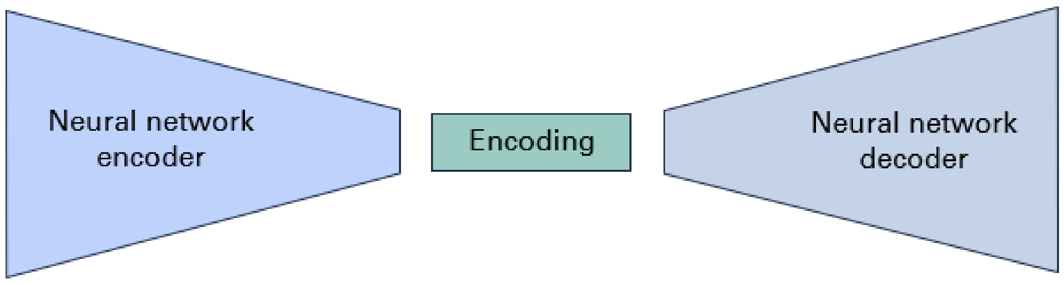


Figure 14. Autoencoder architecture example

The neural network architectures presented in this work resembles the autoencoder. The encoder can be used to perform feature extraction from raw data. However, the decoders implemented in the nets are not either meant to reconstruct the inputs nor to deliver predictions with the same nature of the inputs. Autoencoder is needed in order to process the recordings from the two audio channels, to highlight and conserve important features related to the indoor reflection of an emitted sound and to extract the depth metric values from the audio embedding.

2.3.2 General structure

In this section, a general structure of an ANN in Pytorch is presented.

The artificial neural network implementation is supported by *PyTorch*, machine learning framework chosen in this work among the others (*Keras*, *TensorFlow*, etc.). PyTorch is based on Python programming language. The definition of an ANN model in Pytorch is supported modularly, overwriting standard frame classes, provided in PyTorch libraries. In PyTorch, the neural network architecture is structured in classes in order to sort the single parts of an ANN.

The dataset is provided through the *Dataset* class, or other available dataset classes, that, through its methods, structure the data and supports the *Dataloader()* in data iteration and ANN feeding. All the different architectures developed in this work start with the h5 file reading and data uploading by means of the Dataset class. The *Dataset()* class application requires the overwriting of three main methods: initialization, *.__getitem__(), .__len__().* The first method permits to upload the dataset; the *.__getitem__()* method allows to define which data and in which order are delivered; the last one supports the Dataloader() in iterating over the entire dataset.

The ANN model is uploaded to a CUDA device when available. The method *model.train()* allows to access the training functions and variables; during a Pytorch model training, the Dataloader is accessed at each epoch and provides the data; the net training iterates over them. The net is trained batch-wisely, where a batch is a number of data selected from the original dataset. The batch size, as the number of epochs, loss criterion and other variables, change consistently from case to case, from net to net. Thanks to an *shuffle* option available in the Dataloader(), the coupled samples fed into the NN do not follow the order of the h5py dataset, the nets get sample groups of data in no particular order.

At the beginning of each iteration of the training for loop the optimizer parameters are set to zero and the model provides the prediction from the inputs. Then, the loss is calculated applying the defined criterion and the consequent update of the model parameters is in relation with *loss.backward()*. The optimizer is updated (*optimizer.step()*) and the first training step can be concluded.

The training process and depth results are evaluated qualitatively and quantitatively. The estimation is carried out collecting indicators for each training epoch and considering the trend of the key parameters. The parameters are the training loss, the accuracy, the average of the absolute error between reconstructions and labels calculated on the batch at the current epoch, but also average and mean of the medians obtained from each absolute error that can be obtained in a batch. The training loss trend is plotted, considering the number of epochs in the horizontal axis. The absolute error between reconstruction and label from the batch can be analysed quantitatively, but also envisioned qualitatively: it is shown in a logarithmic histogram and a grayscale picture, complemented by a colour bar for the visualization of the association between grey tone and entity of the error. The other plots that have been approached to support the visualization of the error distribution and entity are the violin plot, the boxenplot and the boxplot. Moreover, the depth results are shown in grayscale and *jet* colormap, accompanied by a colour bar.

If the training phase allows the artificial neural network to correct the parameters and steer them in order to reduce the error provided by the application of the loss criterion to results and corresponding labels, the test helps in analysing the efficacy of the accomplished training. During the test phase, the reconstructions come from a novel dataset fed into the net. The training and parameters evaluation is completed by adding a test phase at each epoch, the validation, that permits to track the progress the net is obtaining at each step and on unseen data.

2.3.3 Loss investigation

The criterion applied in loss calculation consistently affects the performance of an ANN. The criterions presented in this section are the absolute error and mean squared error methods, the most common in audio and video neural networks.

In Pytorch the *L1Loss()* allows to create a criterion to measure the mean absolute error (MAE) between each element in the input *x* and target *y*, considering *x* and *y* tensors of arbitrary shapes with a total of *n* elements each. It is implemented in the NNs presented in this work as unreduced loss, that is described by the following formula: $l(x, y) = L = \{l_1, \dots, l_N\}^T, \quad l_n = |x_n - y_n|$ where *N* is the batch size. To use a criterion that measures the mean squared error between each element in *x* and *y*, the class that should be initialize is the *MSELoss().* It is implemented as the mean reduced version, which corresponds to: $l(x, y) = mean(L), \quad L = \{l_1, \dots, l_N\}^T, \quad l_n = (x_n - y_n)^2$.

However in every architecture in this work where the L1 Loss is selected as the loss criterion, the *.backward()* method is applied to the mean of the loss matrix resulted from the application of the criterion. The reduced method is selected in order to visualize to distribution of the error. The training with loss functions like MSE or L1 loss make the predictions getting closer to the mean value to achieve a low loss. An aspect to consider in relation with the regression loss criterions is the accuracy of the predictions. The gradients of MSE Loss and L1 Loss decrease as the prediction approximates the ground truth. It is generally true for any architecture tested in this work and for the mentioned tasks, that the MSE Loss optimization does not give accurate predictions. L1 Loss, that keeps the gradient large enough when the predicted value is close to the ground truth value, helps optimization and improves the depth predictions.

2.3.4 Optimizer

The optimizer selected for every architecture is Adam. The name Adam is derived from *adaptive moment estimation*. Adam is an optimization algorithm able to update network weights iteratively and based on training data. The Adam optimization algorithm is an extension to stochastic gradient descent that is broadly used for deep learning applications in computer vision and natural language processing.

Gradient descent is an iterative algorithm, its action can be simply explained as starting from a random point on a function and travelling down its slope in steps until it reaches the lowest point of that function. However, the stochastic gradient descent maintains a single learning rate (termed alpha) for all weight updates and the learning rate does not change during training. A learning rate is maintained for each network weight (parameter) and separately adapted as learning unfolds.

The Adam method computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients. The authors describe Adam as combining the advantages of two other extensions of stochastic gradient descent, Adaptive Gradient Algorithm (AdaGrad) and Root Mean Square Propagation (RMSProp). However, instead of adapting the parameter learning rates based on the average first moment (the mean) as in RMSProp, Adam also makes use of the average of the second moments of the gradients (the uncentered variance). It is also straightforward to implement and computationally efficient; little memory is required. The Adam paper suggests good default settings for the tested machine learning problems are alpha=0.001, beta1=0.9, beta2=0.999 and epsilon=10−8. The popular deep learning libraries generally use the default parameters recommended by the paper. The parameters chosen for any application presented in this paper are the standard ones in Pytorch.

2.3.5 Layer investigation

The artificial neural networks implemented in this work are mainly convolutional neural network. In this work, the considered convolutional layers are 2D convolutional layers. A filter or a kernel in a convolutional 2D layer imaginarily slides over the 2D input data, performing an elementwise multiplication and summing up the results into a single output pixel. The kernel performs the same operation for every location it slides over, it can be elementwise or every fixed number of pixels, defined by the strike parameter. Another layer index is the padding, the number of pixels with a fixed value (usually zero) to add at the extremities of the data. As the convolution kernel slides along the input matrix for the layer, the convolution operation generates a feature map, also called activation map, which in turn contributes to the input of the next layer.

The Pytorch NN library provides all the different type of layers. In case of convolutional 2D layers, the input must be a tensor with a shape: (batch size) × (input channels) x (input height) × (input width) or (batch size)

× (input channels) x (input width). This requirement justifies the previous reshaping of the input tensors during the data pre-processing. In a CNN, a convolutional layer can be followed by other layers such as fully connected layers, pooling layers, normalization layers and activation layers. A fully connected layer relates every neuron to every neuron in the next layer. Pooling layers reduce the dimensions of data by combining the outputs of neuron clusters at one layer into a single neuron in the next layer. Max pooling uses the maximum value of each local cluster of neurons in the feature map, while average pooling takes the average value. Normalization layers have the purpose to stabilize the training through the normalization of the input by reshaping and recentralization. The normalization layer largely implemented in this work is the batch normalization, that applies a transformation that maintains the mean output close to 0 and the output standard deviation close to 1. An activation function is the last component of the convolutional layer to increase the non-linearity in the output. Generally, ReLu function or Tanh function is used as an activation function in a convolution layer; however, in this work, only the first mentioned is implemented. The related works are focused on the CNN investigation, since the inputs are images and audio recordings [3-8]; however, the application of fully connected nets can be considered as an alternative [2].

3 Presentation of the neural networks

In this chapter, artificial neural networks are described.

The first paragraph (*Paragraph 3.1*) is dedicated to the presentation of the customized version realized from the article [10]. In the next section, the first network implemented to enhance the performance of the previous network is presented (*Paragraph 3.2*). Alternative solutions for the sub-networks are provided, starting from sound (*Paragraphs 3.3*) and image (*Paragraphs 3.4*) investigation. Then, multiple multi-modality architectures are described in *Paragraph 3.5*. The outperformers are widely discussed in relation to their results in the last section (*Paragraph 3.6*).

In this study, many artificial neural network investigations and versions are evaluated. The guiding line is extracting the embedding of the audio recording and/or of the image sample through the action of an encoder, subsequently a decoder reconstructs the depth map from the latent space. The basic ANN structure under each architecture resembles an autoencoder

A brief list of all the architectures analysed in this work follows.

- *Paragraph 3.1*– Audio modality Neural Network: BatVision
- *Paragraph 3.2* – Audio-video multi-modality Neural Network: Early Convergent AE triplet Net
- *Paragraph 3.3* – Audio encoder investigation: SoundNet, Fully Connected Net and Long Short-Term Memory Net
- *Paragraph 3.4* – Image encoder investigation: VisualEcho, ResNet18, VGG16
- *Paragraph 3.5* – Comparison between different Audio-Video multi-modality neural networks:
    - Early Convergent AE triplet Net
    - Early Convergent AE Net
    - Two Independent Audio - Video Branches Net
    - Two Independent Audio and Audio-Video Branches Net
    - U-Net

3.1 Reference Neural Network

The first investigation is focused on predicting the depth map from indoor audio reverberations. This approach is the starting point of this work and can be found in [10]. The article [10] is profoundly analysed in order to deliver an artificial neural network that follows the concept behind it and implements a customized version of the same architecture. The motivations that conduct to focus on an architecture proposed by an article are, firstly, to evaluate if the article [10] research is reproducible and effective; secondly, the necessity to perform a validation of the dataset. This process is a consequence of the absence of a shared and proven dataset. Moreover, in [10] the data collection setting and the pre-processing are not explained in detail. In the next paragraph, a description of the net follows.

3.1.1 BatVision Net description

The raw audio data can be inputted in the neural network in the time domain or in the frequency domain. In the state-of-art, both the ways can be found, with different results [2-8]. In both the cases, the neural network has to learn how to distinguish the differences in reflected and incident sound, and in energy, through the multiple audible recordings. In this work the approach is utilizing audio data as time series, since the authors of article [10] demonstrate it is the outperforming method, comparing to the use of the spectrogram.

In this work, as explained in *Paragraph 2.2*, the number of time series samples that are taken as input by the neural network to let it consider the entire reverberation phenomenon corresponds to 2200, that is a different length of the input tensor considering the pre-processing presented in [10]. Hence, the encoder is shaped taking into account the peculiarities of the [10] net, however customized for the particular application of this case.

The audio data pre-processing returns two outputs with dimension (1,1,2200) from left and right microphones. This shape allows to operate by means of the convolutional 2D layers, providing a 1D kernel, and avoiding reshaping the embedding afterwards. This configuration is also adopted in [8] and [9] Pytorch code. The first operation is the concatenation of two recordings at the level of the channel dimension. The input (batch size, 2, 1, 2200) is processed through the net.

The *BatVision* neural network is a convolutional neural network that operate with a sequential application of encoding blocks with fixed architecture and different parameters every time. The block is composed of a convolutional layer, a normalization layer (Batch Normalization) and an activation layer (ReLu). The block is repeated eight times: the convolutional layer parameters are presented in the following table.

Block	Layer	# of Filter	Filter size	Padding	Stride
	Conv2D	64	1x228	1x2	0x114
1	BatchNorm2D	64			
	ReLu				
	Conv2D	64	1x128	1x3	0x64
2	BatchNorm2D	64			
	ReLu				
	Conv2D	128	1x64	1x3	0x32
3	BatchNorm2D	128			
	ReLu				
	Conv2D	256	1x32	1x3	0x16
4	BatchNorm2D	256			
	ReLu				
	Conv2D	512	1x16	1x3	0x8
5	BatchNorm2D	512			
	ReLu				
	Conv2D	512	1x8	1x3	0x4
6	BatchNorm2D	512			
	ReLu				
	Conv2D	512	1x4	1x3	0x2
7	BatchNorm2D	512			
	ReLu				
	Conv2D	1024	1x3	1x3	0x1
8	BatchNorm2D	1024			
	ReLu				

Table 1. BatVision Neural Network encoder layer parameters

Similarly, the decoder is developed as a series of six decode blocks and a convolutional layer as the last operation. The decode block reverses the encode block and is composed of a convolutional 2D transposed layer, a batch normalization layer and a ReLu activation layer. The decoder architecture and parameters are summarized in the following table.

Block	Layer	# of Filter	Filter size	Stride	Padding
	ConvTranspose2D	512	4x4	1x1	0x0
1	BatchNorm2D	512			
	ReLu				
	ConvTranspose2D	512	4x4	2x2	1x1
2	BatchNorm2D	512			
	ReLu				

	ConvTranspose2D	256	4x4	2x2	1x1
3	BatchNorm2D	256			
	ReLu				
	ConvTranspose2D	128	4x4	2x2	1x1
4	BatchNorm2D	128			
	ReLu				
	ConvTranspose2D	128	4x4	2x2	1x1
5	BatchNorm2D	128			
	ReLu				
	ConvTranspose2D	64	3x3	1x1	1x1
6	BatchNorm2D	64			
	ReLu				
	ConvTranspose2D	1	1x1	-	-

Table 2. BatVision Neural Network decoder layer parameters

After the concatenation of the two channel recordings, the audio encoder processes the previously normalized audio inputs to return an audio embedding. Then, the depth decoder operates to predict the depth map with metric values from the normalized embedding in the latent space. During the training phase, the results are compared to the corresponding labels to obtain the loss and correct the net model parameters. However, after this learning process, the net can be used as an unsupervised model for depth mapping, given audio recordings from two separate microphones. The BatVision network architecture is shown in the following picture.

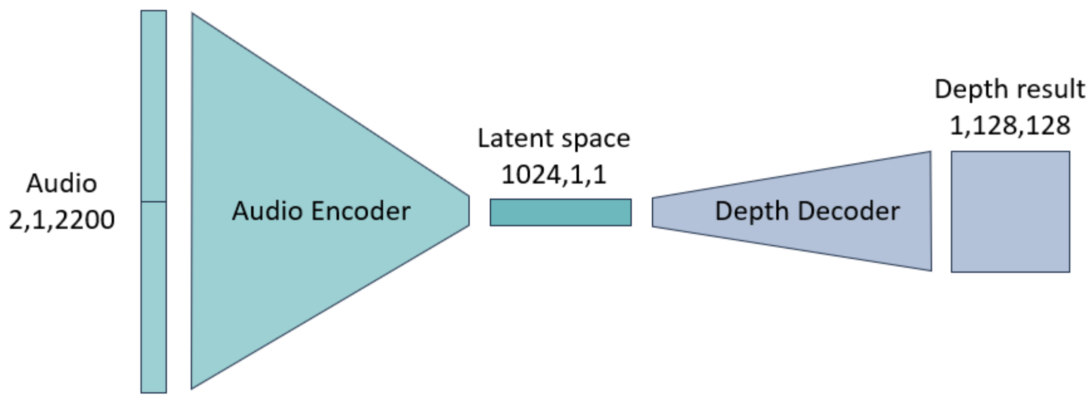


Figure 15. BatVision Neural Network architecture

3.2 Audio – Video Neural Network: Early convergent AE triplet Net architecture

An improvement on the BatVision net performances can be achieved considering the advantages of adding an input with a different nature: images of the scenario, the reverberations of which are gotten by the microphones. Hence, the first ANN investigation naturally moves the attention to the cross-modal implementation and multimodal analysis. The first approach to the image stream introduction in the ANN is an architecture that tries to highlight the nature of the inputs and their descriptors.

The structure resembles two separate autoencoders that process respectively the two channel audio, concatenated as mentioned for the BatVision Neural Network, and the RGB image. The audio and image encodings, both with shape (512, 1, 1), are also concatenated in a tensor with dimensions (1024, 1, 1). This vector is the input of a third decoder that reconstructs the depth in metric values. An L1Loss criterion is assigned to the three error between label and the prediction delivered by every decoder. Despite the main scope is the depth prediction, the intent is to not freeze the two autoencoders and to propagate the error backward from the audio and image reconstructions. The convergency of audio and video data is provided before the depth prediction. The two autoencoders learn to reconstruct the inputs and to avoid corrupting data features and nature during the embedding extraction. The illustration of the architecture is provided in *Figure 16*.

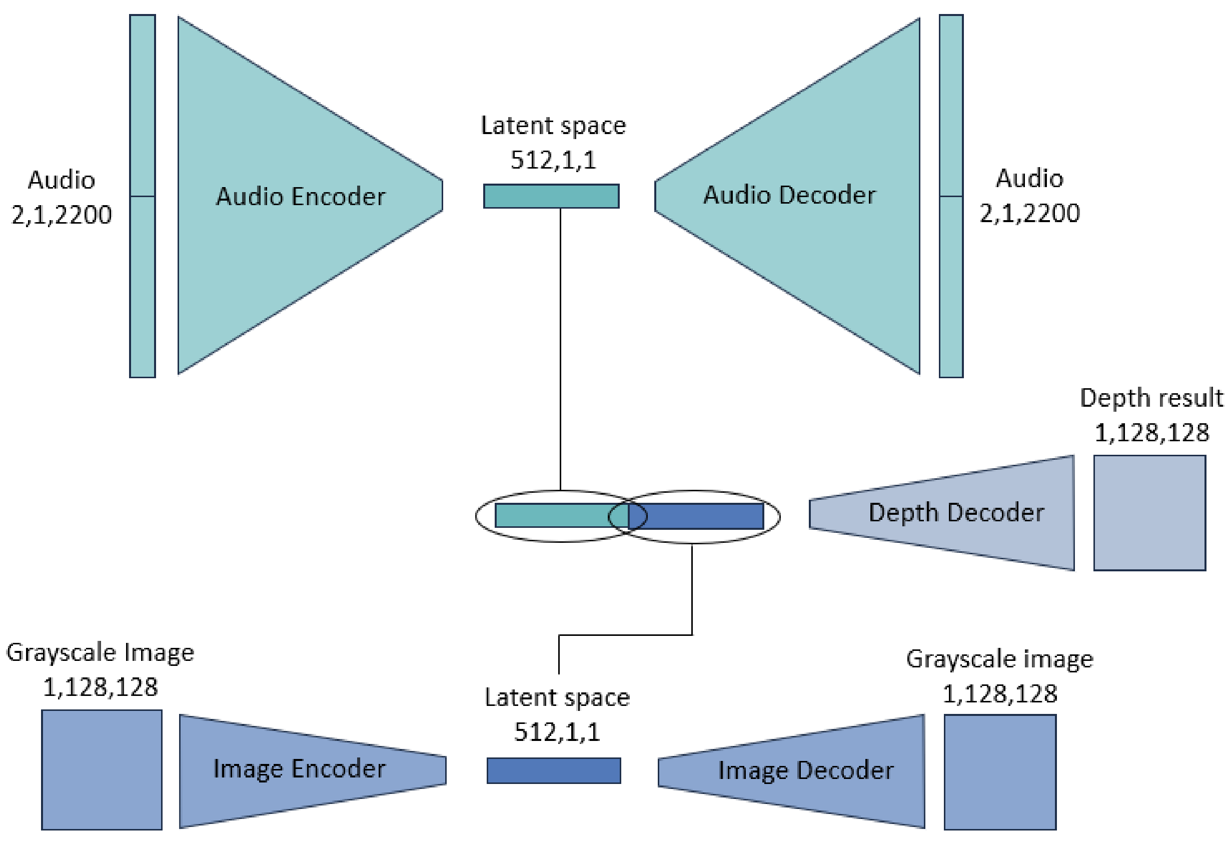


Figure 16. Early convergent AE triplet architecture Neural Network architecture

The audio encoder architecture corresponds to the encoder of the BatVision NN, described above. The audio decoder is obtained by reversing the audio encoder architecture. The depth decoder is taken from the BatVision NN decoder, however it is modified to provide a channel size of encoding of 512. The same shape of audio embeddings (512,1,1) is also assigned to the image embedding. The image encoder has to be reversed. Hence, to facilitate this operation, and in response to the multiple CNN articles to operate with images and to deliver image descriptors vectors, available in the state-of-art [22-24], the encoder is a CNN composed by only convolutional, activation and normalization layers. The parameters of the convolutional layers repropose the audio encoder scheme. The image encoder structure is represented in the table.

Block	Layer	# of Filter	Filter size	Padding	Stride
	Conv2D	30	4x4	2x2	1x1
1	BatchNorm2D	30			
	ReLu				
	Conv2D	60	4x4	2x2	2x2
2	BatchNorm2D	60			
	ReLu				
	Conv2D	120	4x4	2x2	0x0

3	BatchNorm2D	120			
	ReLu				
	Conv2D	240	4x4	1x1	0x0
4	BatchNorm2D	240			
	ReLu				
	Conv2D	512	4x4	1x1	0x0
5	BatchNorm2D	512			
	ReLu				
	Conv2D	512	4x4	1x1	0x0
6	BatchNorm2D	512			
	ReLu				
	Conv2D	512	3x3	1x1	0x0
7	BatchNorm2D	512			
	ReLu				

Table 3. Image encoder layer parameters

The image decoder is obtained by reversing the convolutional layers and assigning convolutional transposed layers.

3.3 Audio Neural Network investigation

The investigations in this section involve audio encoders and the relative predictions. A brief list of the architectures analysed in this section follows.

- SoundNet
- Fully connected Net
- LSTM Net

3.3.1 SoundNet

Even if the results delivered by the customized BatVision version are acceptable, the parallel image investigation, presented in the next paragraph, highlights that inserting Max Pooling layers means maintaining relevant features for the task of depth predicting.

BatVision Audio Neural Network relies on a CNN that applies only convolutional layers, excluding MaxPooling layer application. However, a popular paper [8] provides two versions of Audio NN for audio feature extraction and mainly applied in sound classification: the SoundNet eight layers version involves the MaxPooling application. Additionally, in image feature extraction its implementation results to have a fundamental positive contribution, discussed in the next session. The convolutional layer has the property to smooth and apply an averaging action, as mentioned in *Paragraph 2.3.5*; the depth reconstructions from the embeddings come to get significant with a considerable amount of training data in this application, more than five thousand of data, like the acquired dataset. The MaxPooling layer implementation allows to highlight significant features, that are delivered and conserved in the embeddings.

The SoundNet eight layers version is customized to be applied in this research, mostly to be in agreement with the input dimensions and embedding size suitable for working out with the depth decoder. The customized version is a sequential application of a convolutional 2D layer, a batch normalization layer (BatchNorm2D) and an activation layer (ReLu). The presence of a MaxPooling 2D layer is positioned after the first and the second block, at the very early stages, as summarized in *Table 4*.

Block	Layer	# of Filter	Filter size	Stride	Padding
	Conv2D	16	64x1	2x1	32x0
1	BatchNorm2D	16			
	ReLu				
	MaxPool2D	16	8x1	8x1	

	Conv2D	32	32x1	2x1	16x0
2	BatchNorm2D	32			
	ReLu				
	MaxPool2D	32	8x1	8x1	
	Conv2D	64	16x1	2x1	8x0
3	BatchNorm2D	64			
	ReLu				
	Conv2D	128	8x1	2x1	4x0
4	BatchNorm2D	128			
	ReLu				
	Conv2D	256	4x1	2x1	2x0
5	BatchNorm2D	256			
	ReLu				
	MaxPool2D	256	2x1	2x1	
	Conv2D	512	4x1	2x1	2x0
6	BatchNorm2D	512			
	ReLu				
	Conv2D	512	4x1	2x1	2x0
7	BatchNorm2D	512			
	ReLu				

Table 4. SoundNet Neural Network layer parameters

The SoundNet NN is implemented as the encoder of the AE architecture. The decoder is obtained replicating the BatVision decoder and utilizing the same parameters. The overall architecture resembles an autoencoder. The complete architecture is shown in *Figure 17*.

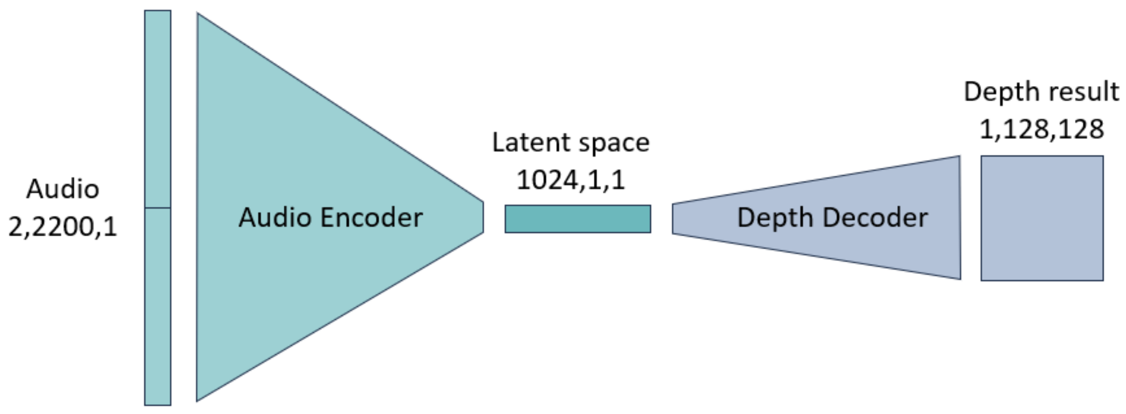


Figure 17. SoundNet Neural Network architecture

3.3.2 Fully connected Net

The investigation of a fully connected version for the audio encoder is proposed. The main reason is in relation to the audio input nature, a time series. Taking matrix vectors, like images, as input justify the use of Convolutional 2D layers. Despite many of the articles, presented in the references, implement the CNN also for audio feature extraction, the state-of-art proposes a various range of possible architecture for time series processing. The employment of a fully connected neural network for Sound Source Localization can be found in [2], that takes a different path compared to the other papers in relation to artificial neural networks for audio application.

In fully connected layers, the neuron applies a linear transformation to the input vector through a weights matrix (*W*). A non-linear transformation is then applied to the product through a non-linear activation function *f*.

$$y_{jk}(\mathrm{x}) = f\left(\sum_{i=1}^{n_H} w_{jk}x_i + w_{j0}\right)$$

Where *x* is the input vector, *W₀* the bias term.

The proposed architecture is a simple series of three fully connected layers, alternated with activation layers (ReLu). The architecture is resumed in *Table 5*.

Block	Layer	# of Filter
1	Linear	2060
	ReLu	
2	Linear	1030
	ReLu	
3	Linear	512
	ReLu	

Table 5. Fully connected Neural Network layer parameters

3.3.3 LSTM Net

LSTM is a type of recurrent neural network (RNN). RNNs are a powerful type of artificial neural network that can internally maintain memory of the input. This makes them particularly suited for solving problems involving sequential data like a time series. It is used for time-series data processing, prediction, and classification. The LSTM model is implemented in [7] for SSL in outdoor environments. It can be considered as one of the most advanced models to calculate time series is the Long Short-Term Memory (LSTM) Neural Network.

LSTM has feedback connections, unlike conventional feed-forward neural networks. It can handle not only single data points (like photos) but also complete data streams (such as speech or video). LSTM cells learn to predict the future from sequences of variable lengths.

The Pytorch LSTM implementation can be realized through *Torch.LSTM().* Its application is followed by a sequence of linear layers and activation layers (ReLU). The Pytorch version offers flexibility in the number of LSTM layers to apply. The starting approach is executing sixty-four LSTM layers, then eight layers.

3.4 Image Neural Network investigation

The Image Neural Network investigation complements the Audio NN research. This section has the purpose of describing the image encoder investigation, implemented in the depth prediction autoencoder. This analysis is conducted in order to optimize the image feature extraction, however all the image encoders presented in this section are also implemented in the Audio-Video Neural Network investigation.

This research goes into the direction of [25-29], to evaluate the depth prediction that results from monocular images. However, the main purpose is comparing the different architecture behaviours in image feature extraction for depth prediction.

The net is fed by RGB images. The image encoder extracts features, that are delivered from the latent space to the decoder. It delivers a depth prediction which is compared to the label, as presented in the previous Audio Neural Networks. The decoder is the same of the previous nets. The architecture is illustrated in the following figure.

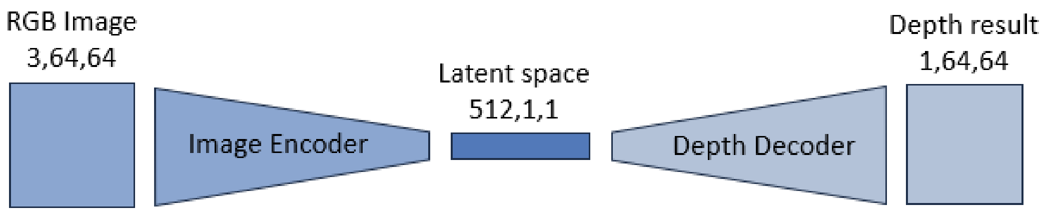


Figure 18. Image Neural Network architecture

The artificial neural networks considered in this paragraph are customized version of ResNet 18, VGG 16 [38] and Visual Echoes net from [9]. These networks are valuable attempts to apply CNN encoders, that involve pooling layers, to evaluate how this layer can affect the results.

3.4.1 ResNet 18

The ResNet reference is the article [32]; this net is mentioned in the research presented in [4], where it is applied for image feature extraction in audio-visual scene analysis. This net is customized for the application, however it is also implemented with the pre-trained weights proposed by Pytorch. The architecture of the ResNet 18 customized version is summarized in the following table.

Block	Layer	# of Filter	Filter size	Stride	Padding	Output
1	Conv2D	64	7x7	2x2	3x3	
	MaxPool2D	64	3x3	2x2	1x1	
	BatchNorm2D	64				
	ReLu					
2	Conv2D	64	3x3	1x1	1x1	
	BatchNorm2D	64				
	ReLu					
	Conv2D	64	3x3	1x1	1x1	
	BatchNorm2D	64				
	ReLu					
	ReLu					
	Conv2D	64	3x3	1x1	1x1	
	BatchNorm2D	64				
	ReLu					
	Conv2D	64	3x3	1x1	1x1	
	BatchNorm2D	64				
	ReLu					
	ReLu					
3	Conv2D	128	1x1	2x2	0x0	
	Conv2D	128	3x3	2x2	1x1	
	BatchNorm2D	128				
	ReLu					A
	Conv2D	128	3x3	1x1	1x1	
	BatchNorm2D	128				
	ReLu					B
						a+b
	Conv2D	128	3x3	1x1	1x1	
	BatchNorm2D	128				
	ReLu					
	Conv2D	128	3x3	1x1	1x1	
	BatchNorm2D	128				
	ReLu					
	ReLu					
4	Conv2D	256	1x1	2x2	0x0	
	Conv2D	256	3x3	2x2	1x1	
	BatchNorm2D	256				
	ReLu					A
	Conv2D	256	3x3	1x1	1x1	
	BatchNorm2D	256				

	ReLu					B
						a+b
	Conv2D	256	3x3	1x1	1x1	
	BatchNorm2D	256				
	ReLu					
	Conv2D	256	3x3	1x1	1x1	
	BatchNorm2D	256				
	ReLu					
	ReLu					
5	Conv2D	512	1x1	2x2	0x0	
	Conv2D	512	3x3	2x2	1x1	
	BatchNorm2D	512				
	ReLu					A
	Conv2D	512	3x3	1x1	1x1	
	BatchNorm2D	512				
	ReLu					B
						a+b
	Conv2D	512	3x3	1x1	1x1	
	BatchNorm2D	512				
	ReLu					
	Conv2D	512	3x3	1x1	1x1	
	BatchNorm2D	512				
	ReLu					
	ReLu					
	AdaptiveAvgPool2D		1x1			

Table 6. ResNet 18 Neural Network layer parameters

3.4.2 VGG 16

Some of the most famous ANN produced to work with and on images are the VGG CNN versions. The VGG investigation comes up with [3, 5, 6]. VGG 16 is presented in the paper [38]. The authors ensure their net can work out with many different datasets, showing that it achieves state-of-the-art results.

The VGG 16 implementation in this work is gotten by adding two more block at the end, to achieve the requested shape of the embedding. The VGG 19 complete architecture is insufficient, as well. The VGG16 is also used as a pretrained model utilizing the VGG16 pretrained weights provided by Pytorch. The architecture of the customized version is represented in *Table 7*.

Block	Layer	# of Filter	Filter size	Stride	Padding
	Conv2D	64	3x3	1x1	1x1
	BatchNorm2D	64			

	ReLu				
1	Conv2D	64	3x3	1x1	1x1
	BatchNorm2D	64			
	ReLu				
	MaxPool2D	128	2x2	2x2	0x0
	Conv2D	128	3x3	1x1	1x1
	BatchNorm2D	128			
	ReLu				
2	Conv2D	128	3x3	1x1	1x1
	BatchNorm2D	128			
	ReLu				
	MaxPool2D	128	2x2	2x2	0x0
	Conv2D	256	3x3	1x1	1x1
	BatchNorm2D	256			
	ReLu				
	Conv2D	256	3x3	1x1	1x1
3	BatchNorm2D	256			
	ReLu				
	Conv2D	256	3x3	1x1	1x1
	BatchNorm2D	256			
	ReLu				
	MaxPool2D	256	2x2	2x2	0x0
	Conv2D	512	3x3	1x1	1x1
	BatchNorm2D	512			
	ReLu				
	Conv2D	512	3x3	1x1	1x1
4	BatchNorm2D	512			
	ReLu				
	Conv2D	512	3x3	1x1	1x1
	BatchNorm2D	512			
	ReLu				
	MaxPool2D	512	2x2	2x2	0x0
	Conv2D	512	3x3	1x1	1x1
	BatchNorm2D	512			
	ReLu				
	Conv2D	512	3x3	1x1	1x1
5	BatchNorm2D	512			
	ReLu				
	Conv2D	512	3x3	1x1	1x1
	BatchNorm2D	512			
	ReLu				
	MaxPool2D	512	2x2	2x2	0x0
	Conv2D	512	3x3	1x1	1x1
	BatchNorm2D	512			
	ReLu				

	Conv2D	512	3x3	1x1	1x1
6	BatchNorm2D	512			
	ReLu				
	Conv2D	512	3x3	1x1	1x1
	BatchNorm2D	512			
	ReLu				
	MaxPool2D	512	2x2	2x2	0x0

Table 7. VGG 16 Neural Network layer parameters

3.4.3 VisualEcho

This net is a CNN presented in the article [11], which implemented this network as a branch for image feature extraction in a wider net for depth prediction. Since it is designed for a similar task, it is implemented and tested in the architectures of this work. It is characterized by a well-organized architecture that can be explained with a block sequence. Its implementation is customised for this application, in respect with the image embedding needed dimensions. Its architecture is shown in the next table.

Block	Layer	# of Filter	Filter size	Stride	Padding
	Conv2D	64	3x3	2x2	1x1
1	BatchNorm2D	64			
	ReLu				
	MaxPool2D	64	2x2	2x2	0x0
	Conv2D	128	3x3	1x1	1x1
2	BatchNorm2D	128			
	ReLu				
	MaxPool2D	128	2x2	2x2	0x0
	Conv2D	256	3x3	1x1	1x1
3	BatchNorm2D	256			
	ReLu				
	MaxPool2D	256	2x2	2x2	0x0
	Conv2D	512	3x3	1x1	1x1
4	BatchNorm2D	512			
	ReLu				
	MaxPool2D	512	4x4	1x1	0x0

Table 8. VisualEcho Neural Network layer parameters

3.5 Audio – Video Neural Network investigation

This chapter is dedicated to cover the most significant architectures involved in the audio-video multi-modality investigation conducted in this thesis. Working with artificial neural network often means facing many different possibilities and versions: not the entire range of versions is going to be covered in the report.

A brief list of the architectures analysed in this section follows.

- Early Convergent AE Net
- Two independent audio - video branches Net
- Two independent audio - audio+video branches Net
- U-Net

3.5.1 Early Convergent Audio – Video Autoencoder Net

The architecture of this network can be visualized in *Figure 19.*

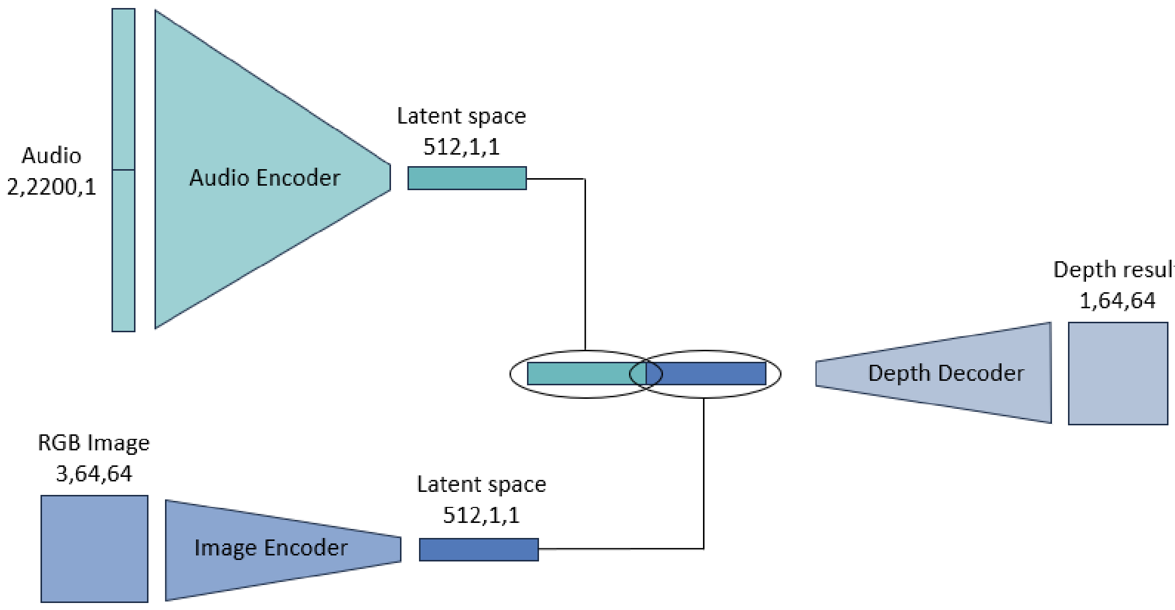


Figure 19. Early Convergent Audio - Video Autoencoder Neural Network architecture

It is composed by two subnetworks. Two of them observe the typical encoder structure: one is related to audio feature extraction, the other one to the image. The audio and image embeddings are then concatenated and taken as input by the decoder. The function of the decoder is predicting the depth related to the input vector and deliver a depth matrix in metric values.

The model is close to the second one proposed in *Paragraph 3.3*, the Early convergent AE triplet architecture deprived of audio and image decoder. The investigation steers on this network, because of comparable results from the two networks but with a bigger computational effort requested by the Early convergent AE triplet net.

In the following investigation, the model audio encoder is set as the SoundNet presented in *Paragraph 3.3.1*; all the Image NNs, described in *Paragraph 3.3.2,* are applied and tested as image encoder in this network. The audio and image encoders are combined to have different versions of this network:

- SoundNet and ResNet 18
- SoundNet and VGG 16
- SoundNet and VisualEcho

3.5.2 Independent Audio and Video Branches Net

In opposition with an early convergency of the multi-modal embeddings presented in the Early Convergent Audio – Video Autoencoder Net, this net allows audio and image inputs to walk through two separate paths. Two different autoencoder process the two inputs separately: audio and image embeddings, from the corresponding encoders, are delivered to two separate decoders to deliver two depth predictions at the same time and independent from each other. The depth predictions are forced to converge as the last step, applying a MSE Loss criterion to the comparison between the two separated reconstructions. Both the autoencoders are trained separately applying the absolute value criterion as the loss criterion, that demonstrates to work successfully for both the single autoencoders in previous investigations. The additional loss is applied to educate the sub-models after the comparison. As mentioned before, the chosen criterion is the mean squared error and it is supposed to smoothen the high divergency of the two reconstructions. This network resembles a late convergency, in contrast with the Early Convergent Audio – Video Autoencoder Net.

Since both the modalities have showed up their advantages and disadvantages, their potentialities and downsides, comparing reconstructions from the same group of input that have followed separate paths can mean highlight the advantages of both the modal application and cure the issues.

The delivered reconstructions come from the audio branch; however, they are previously compared to twin from the image branch in order to correct the depth results by means of the predictions from image path.

The net output is a depth prediction delivered from echolocation and steered by the monocular RGB reconstruction. However, both the nets are trained separately applying the absolute value criterion as the loss criterion, that demonstrates to work successfully for both the single autoencoders. The additional loss is applied to educate the sub-models after the comparison. As mentioned before, the chosen criterion is the mean squared error and it is supposed to smoothen the high divergency of the two reconstructions.

The architecture implemented as the sub-model is the SoundNet NN as audio encoder, followed by the BatVision decoder. For the image descriptor extraction, some experiments are conducted switching from VGG 16 and ResNet 18, with the VGG 16 as the outperformer; the decoder is still the BatVision decoder.

The overall architecture is shown below.

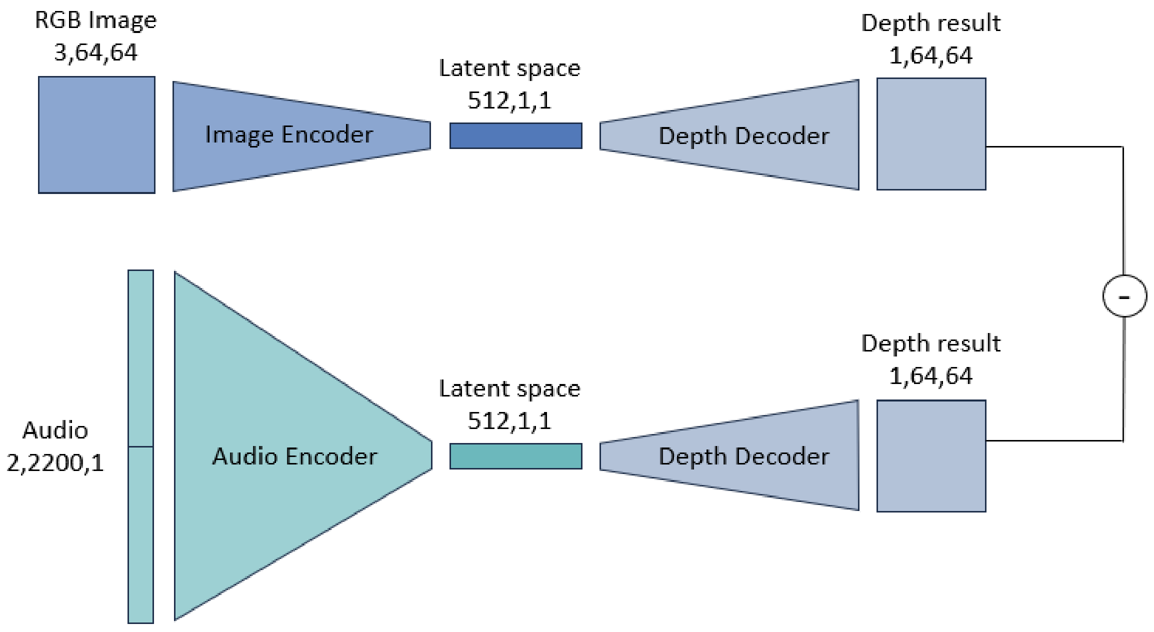


Figure 20. Independent Audio and Video Branches Neural Network architecture

3.5.3 Independent Audio and Audio – Video Branches Net

Since the previous model worked successfully, the next idea is to educate the depth reconstructions that come up from the audio encoder by means of a comparison to the results from the early-convergent multi-modality model presented before.

The architecture components are the same for the audio branch. The other path is the Early Convergent Audio – Video Autoencoder Net, SoundNet and VGG 16 version.

The complete architecture can be visualized in *Figure 20*.

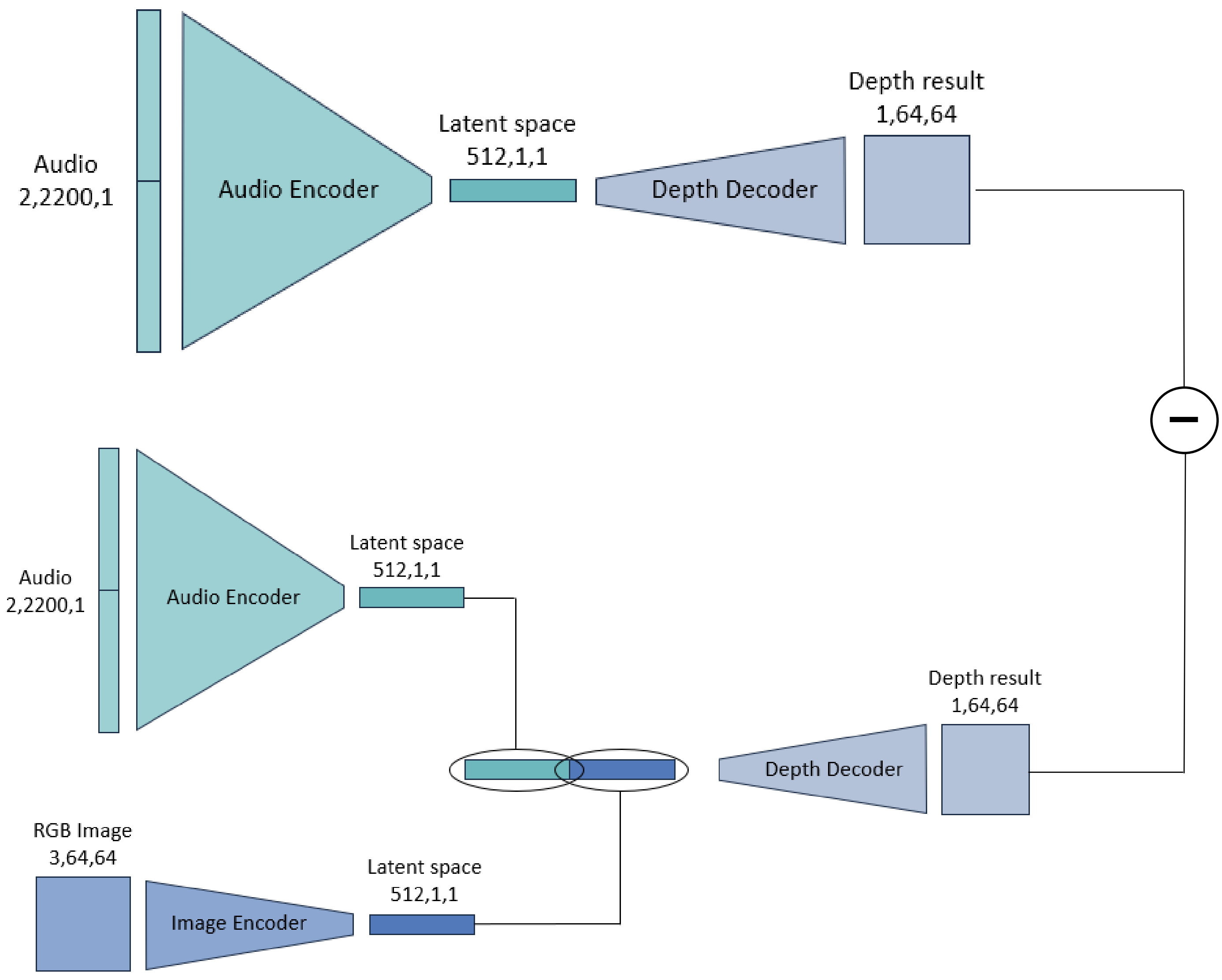


Figure 21. Independent Audio and Audio - Video Branches Neural Network architecture

3.5.4 U-Net

U-Net is a convolutional neural network developed as semantic segmentation technique and originally proposed for medical imaging segmentation. It's one of the earlier deep learning segmentation models, and the U-Net architecture is also used in many GAN variants. Its architecture can be broadly thought of as an encoder network followed by a decoder network.

The network is based on the fully convolutional network and its architecture was modified and extended to work with fewer training images and to yield more precise segmentations. The main idea of the encoder part is to supplement a CNN diminishing network by successive layers. Pooling layers consistently increase the resolution of the output; convolutional layers can learn to assemble a precise output based on this information. The U-Net is named by the path that the data follows: it firstly goes through an encoding part, in which it is downsampled, and then it is upsampled in the second side symmetrically to form a path with a "U" shape. The difference between U-Nets and autoencoders mainly relies on the skip connections, relations between corresponding encoding and decoding layers that can be found only in the U-Nets.

The architecture can be visualized in the successive figure.

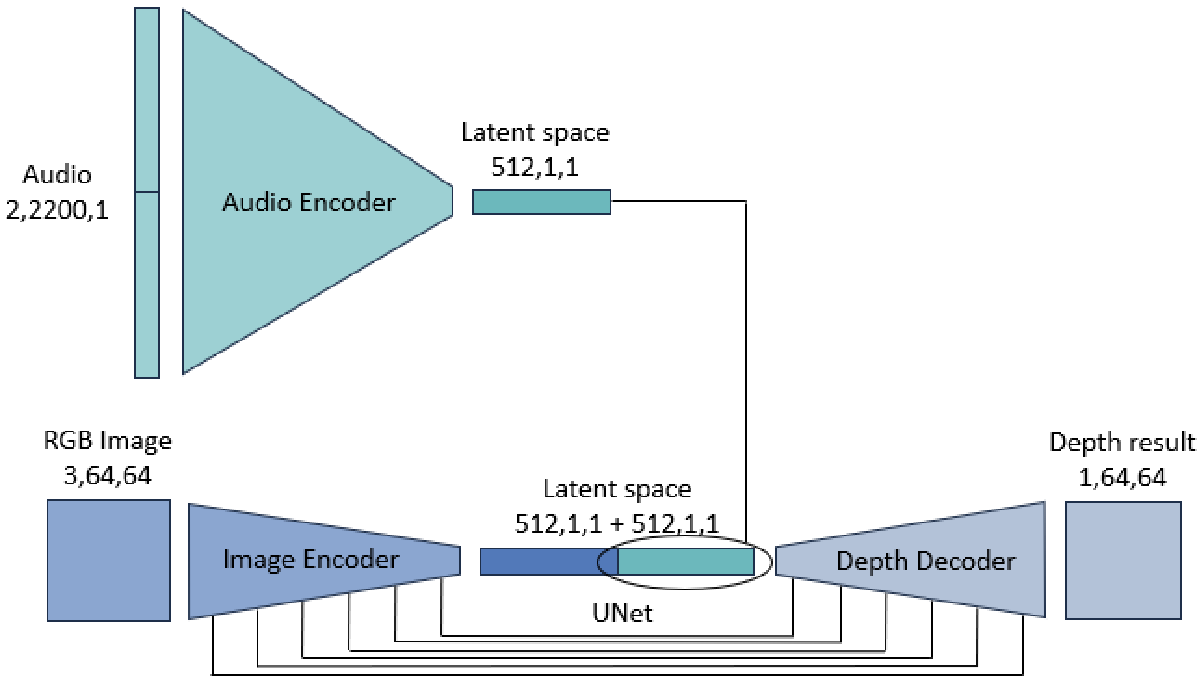


Figure 22. U-Net Neural Network architecture

The net parameters are declared in *Table 11*.

Block	Layer	# of Filter	Filter size	Stride	Padding	Output
	Conv2D	64	3x3	1x1	1x1	
	BatchNorm2D	64				
	ReLu					
1	Conv2D	64	3x3	1x1	1x1	
	BatchNorm2D	64				
	ReLu					x1
	MaxPool2D	64	2x2	2x2		
	Conv2D	128	3x3	1x1	1x1	
2	BatchNorm2D	128				
	ReLu					x2

	MaxPool2D	128	2x2	2x2		
	Conv2D	256	3x3	1x1	1x1	
3	BatchNorm2D	256				
	ReLu					x3
	MaxPool2D	256	2x2	2x2		
	Conv2D	256	3x3	1x1	1x1	
4	BatchNorm2D	256				
	ReLu					x4
	MaxPool2D	256	2x2	2x2		
	Conv2D	512	3x3	1x1	1x1	
5	BatchNorm2D	512				
	ReLu					x5
	MaxPool2D	512	2x2	2x2		
	Conv2D	512	3x3	1x1	1x1	
6	BatchNorm2D	512				
	ReLu					x6
	MaxPool2D	512	2x2	2x2		
	Conv2D	512	3x3	1x1	1x1	
bottle-neck	BatchNorm2D	512				
	ReLu					output
	Concat:	output + SoudNet embedding				
	ConvTranspose2D	512	4x4	2x2	1x1	
6	Conv2D	512	3x3	1x1	1x1	
	BatchNorm2D	512				
	ReLu					output
	Concat:	output + x6				
	ConvTranspose2D	512	4x4	2x2	1x1	
5	Conv2D	512	3x3	1x1	1x1	
	BatchNorm2D	512				
	ReLu					output
	Concat:	output + x5				
	ConvTranspose2D	256	4x4	2x2	1x1	
4	Conv2D	256	3x3	1x1	1x1	
	BatchNorm2D	256				
	ReLu					output
	Concat:	output + x4				
	ConvTranspose2D	256	4x4	2x2	1x1	
3	Conv2D	256	3x3	1x1	1x1	
	BatchNorm2D	256				
	ReLu					output
	Concat:	output + x3				
	ConvTranspose2D	128	4x4	2x2	1x1	
2	Conv2D	128	3x3	1x1	1x1	
	BatchNorm2D	128				

	ReLu					output
	Concat:	output + x2				
	ConvTranspose2D	64	4x4	2x2	1x1	
1	Conv2D	1	3x3	1x1	1x1	

Table 9. U-Net Neural Network layer parameters

3.6 Experiments

This section contains results, evaluation and additional investigation of the networks presented in the previous chapter.

3.6.1 Dataset validation

This paragraph presents the *Dataset 1* and *Dataset 2* comparison showing the quantitative results obtained from their application as input to the BatVision Neural Network.

The dataset validation is needed in order to evaluate the suitability of data collected for this task. The dataset evaluated are *Dataset 1* and *Dataset 2*, presented in Paragraph 2.2.1. The method adopted to evaluate the datasets is the comparison of the validation reconstructions obtained from the two different model of the BatVision Neural Network trained by applying the datasets separately. The data that can be tested are the audio and depth samples, because the article [10] presents a supervised artificial neural network that emulates the echolocation from audio reverberations, without vision data.

The results are evaluated quantitatively, considering the absolute error between reconstructions and labels obtained during the validation at different training epochs. The absolute error is in the normalized space. The histogram illustrated below shows a comparison between the averages of the absolute error calculated between one hundred reconstructions and their corresponding labels, obtained during the validation of the BatVision Net trained separately with Dataset 1 and Dataset 2 and for a diverse number of epochs.

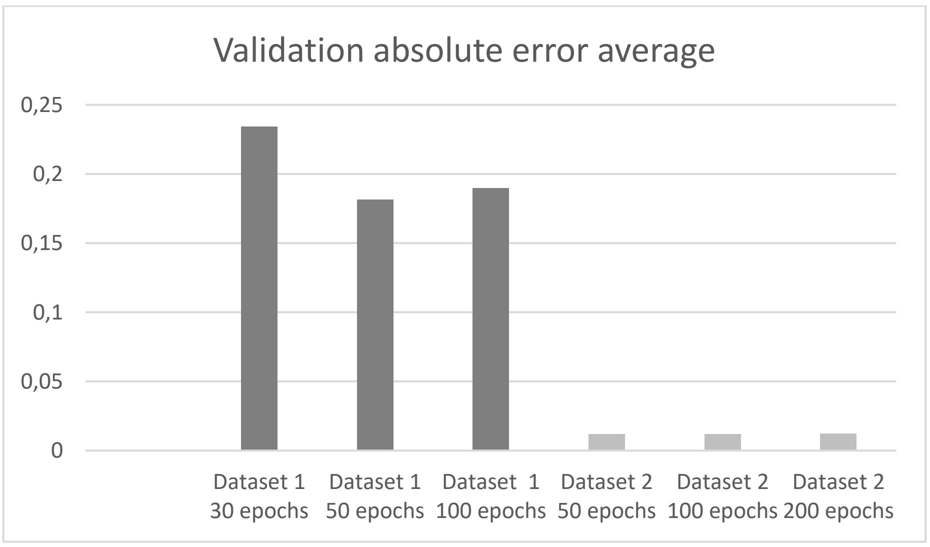


Figure 23. Averages of 100 validation examples for each version.
With the purpose of validating the two successive datasets collected, the article network was trained and

tested with both. In the attempt to be independent to the overfitting issues, the training was performed maintaining the same batch size but changing the number of epochs.

Dataset 2 demonstrates overall a better performance than the first, as shown in *Figure 23*. The central reason is that for the *Dataset 1* the differentiation was provided mainly taking care of moving objects and not steering the sensors setup to change the geometry of the captured portion of the room. However, providing a wider variety at level of room architectures and wall positions affects the audio reverberations in a tangible way for the artificial neural networks (for more detail *Appendix A*). This means that this network works better recognizing if the system is a different room, than sensing the presence of an object in a scenario already seen. Moreover, the dataset differentiation should be added, for instance, acquiring data in different rooms more than sequentially moving the sensor setup along a fixed direction in a certain scenario. A larger differentiation of a dataset is generally recommendable in order to train a neural network.

3.6.2 Early convergent AE triplet Net evaluation

This section is composed by the investigations accomplished working on the network mentioned in the title to improve the performances. The first paragraph is dedicated to the audio encoder: different versions obtained customizing the BatVision Net (*Paragraph 3.1*) are tested in autoencoders for audio data reconstruction. The next paragraph is related to the image investigation; the image encoders are tested in the overall network.

3.6.2.1 Audio reconstruction investigation

Given that audio reconstruction is one of the Early convergent AE triplet Net purposes, different autoencoder versions come from the implementation of multiple BatVision article encoder customizations and corresponding audio decoders resulting from the encoder reverse. The customized version of the audio encoder of the article replicates the BatVision encoder structure presented before. The other versions decrease the number of encoding blocks from six to five (*First version*), four (*Second version*) and three (*Third version*) respectively; the reduction of the layers would correspond to a decreased complexity in audio feature processing, and maybe a better reconstruction both for depth images and audio data. The performances are compared in union with a fully connected version with three layers (*Fourth version*) and the application of all the nets is accomplished with the use of the same dataset and parameters, training for twenty-five epochs.

The following pictures show the training loss trend of different audio autoencoder versions coming from the BatVision encoder customization and the histograms of the average and standard deviation of the absolute errors between labels and reconstructions calculated during the validation and applying the same conditions.

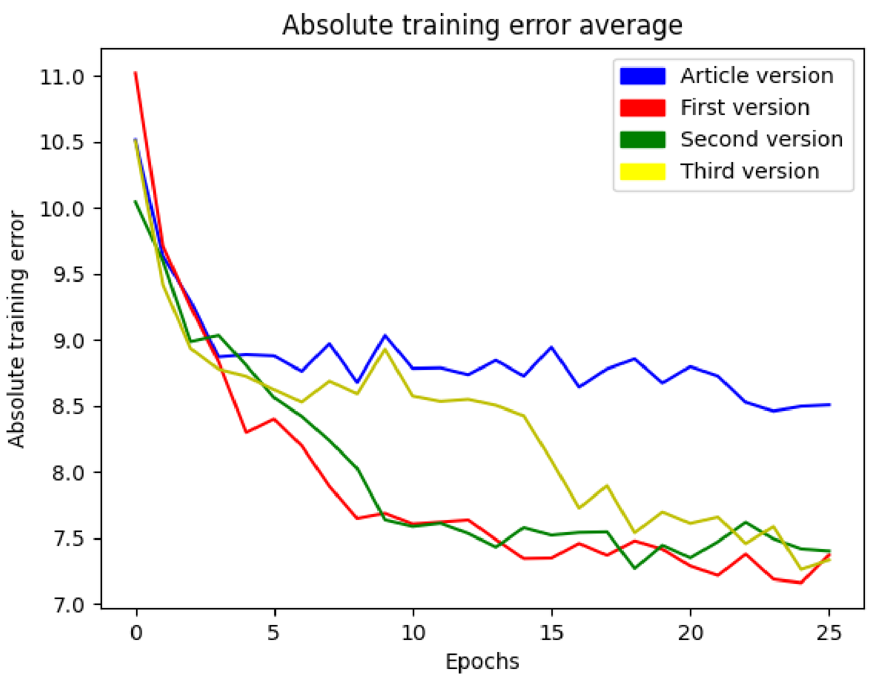


Figure 24. Training loss trend of different audio autoencoder versions coming from the BatVision encoder customization

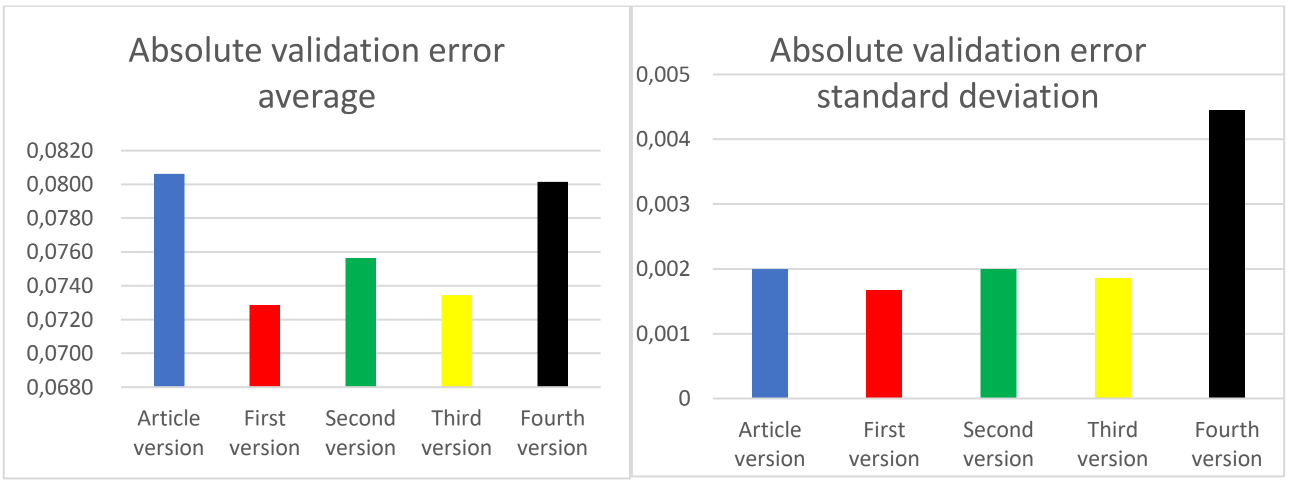


Figure 25. Histograms of the average and standard deviation of the absolute errors calculated in all the evaluation samples delivered through a batch, in same condition of number of epochs, dataset choice, loss and optimizer criterions

The fully connected version, in black, results to be slower in learning; the performance indicators, resumed in training loss trend and average and standard deviation applied to the absolute errors in a validation batch, show diverse performances in audio reconstruction. The next step is founding if it is possible to formulate a correlation between the feature extraction for audio reconstruction and the audio feature extraction for depth prediction. The implementation of all the different versions is tested in the overall architecture to deliver the optimum.

3.6.2.2 Image reconstruction investigation

The image feature extraction realization affects the quality of the depth reconstructions in the Early convergent AE triplet Net.

The first autoencoder version is a simple convolutional encoder-decoder with a fixed kernel dimension of four. This structure can be found around in many articles [43-45]. Another common sequence is a series of convolutional layers and max pooling downsampling layers, like in [24, 46, 47] where the purpose is denoising. However, the pooling has no opposite layer. Also, downsampling is not needed since the imputed images are scaled in resolution.

The second version is an autoencoder with half of the number of layers, four for both encoder and decoder instead of a complex amount of sixteen layers. The convolutional layers have 16x16, 8x8, 4x4, 2x2 kernels in order. Each convolutional layer is followed by BatchNormalization and ReLu layers, as before.

The training reconstructions after the 50th epoch appear acceptable.

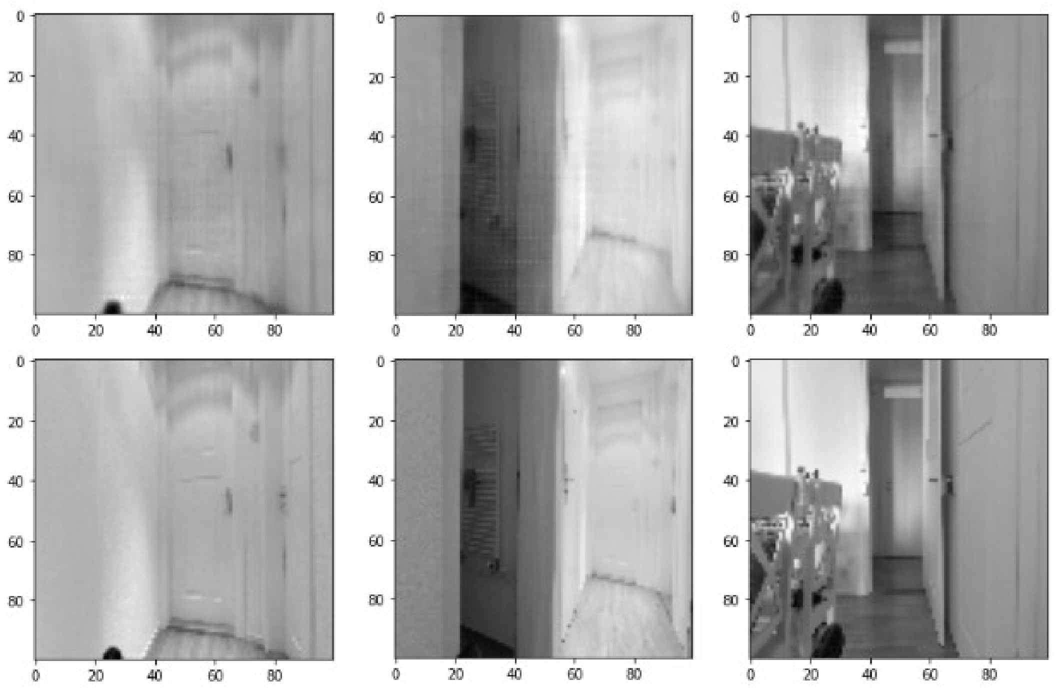


Figure 26. Image reconstructions during the training phase.
The results are shown above the corresponding ground truth.

The images look denoised and are close to the original ones. The training dataset is not larger enough to perform correct validation reconstructions.

After the epoch 30 the training and validation trends appear to start to diverge. It is also reflected in the results.

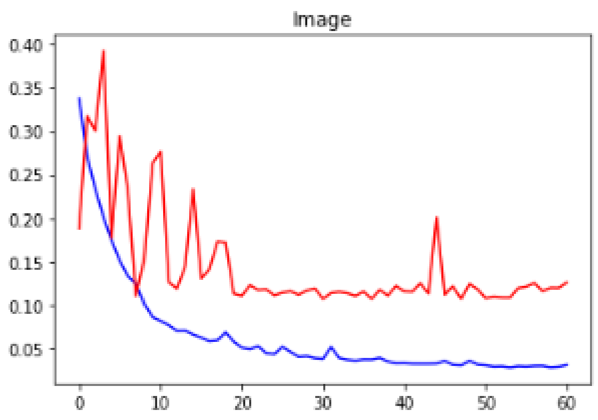


Figure 27. Training (blue line) and validation (red line) loss trends of the image autoencoder

Some validation reconstructions after epoch 50 and the corresponding labels re provided in the next figures.

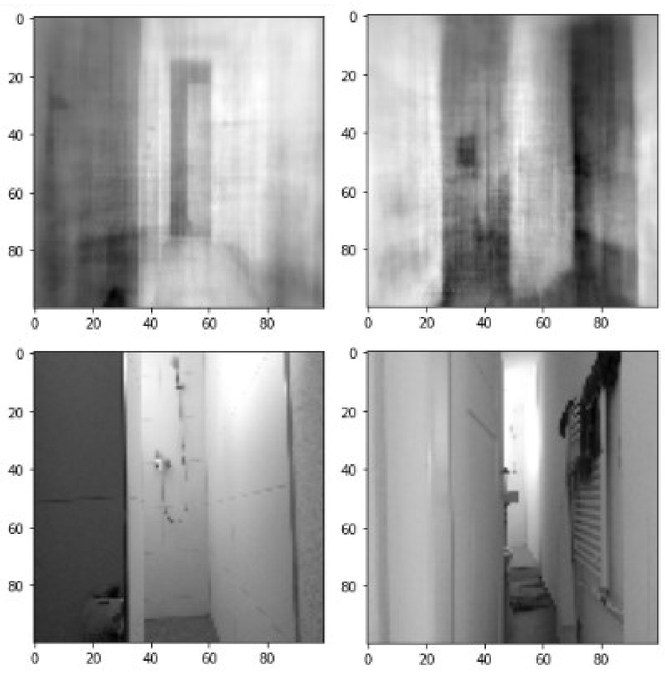


Figure 28. Some exemplification of the validation reconstructions of the Early Convergent AE triplet Net image autoencoder. The results are shown above the corresponding ground truth.

Despite the image reconstructions performed during the training perfectly match the labels, the image results of the test phase are not as pleasant. It is possible to see that in the reconstructions in *Figure 26* the net just proposes a training image. To improve the performance of the net, the image autoencoder is tested separately, also switching the loss calculation from the absolute error criterion (L1 Loss) to mean squared error criterion (MSE Loss). This approach prefers not to force the image convergency, since the image reconstruction is not the main goal of the ANN implementation. However, the quality of the predictions does not benefit evidently from this application.

Despite some good depth reconstructions obtained from the Early Convergent AE triplet Net after the application of previously explained changes, it is clear that the image embeddings affect not in a positive way the depth predictions (*Figure 28*) and the bad performance on image reconstruction. The source of this issue is found in the image feature extraction by the image autoencoder, that results not fully appropriate for this case.

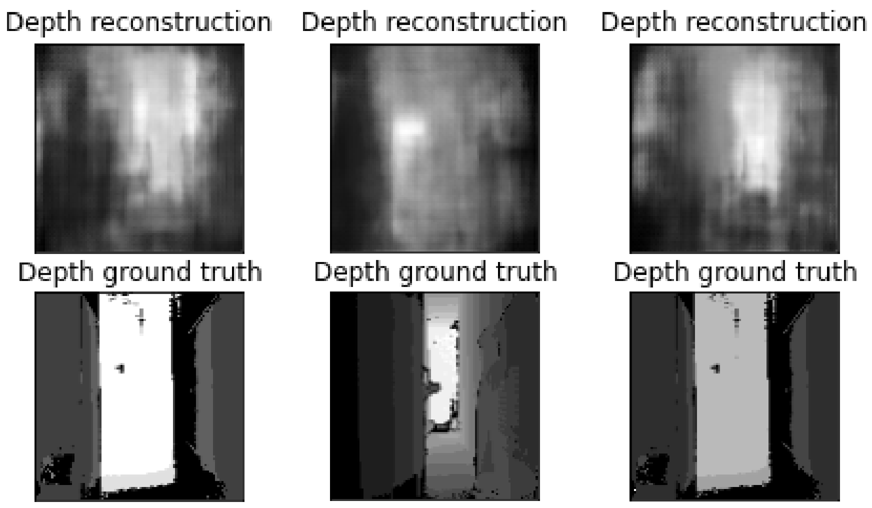


Figure 29. Good depth reconstruction examples.

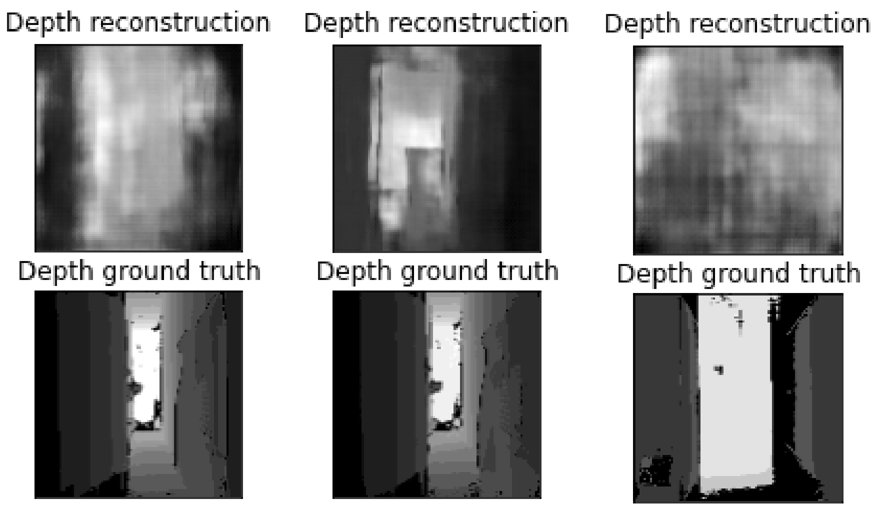


Figure 30. Poor depth reconstruction examples.

This net is applied also in RGB image stream case, by switching the number of channels from one to three. The delivered results confirm the previous conclusions.

Since the reconstructions closely relies on both the embeddings, the encoder importance and proper feature extraction is central. From this new starting point, the ANN examination continues looking at the state-of-art [8, 9, 11] and implementing new strategies and architectures.

3.6.3 Performance comparison

This section is dedicated to covering the most significant architectures in audio-video multi-modality investigation in this work. Working with artificial neural networks often means facing many different possibilities and versions, changing parameters multiple times and continuous testing. Not the entire range of versions is going to be covered in the report: this paragraph is dedicated to proposing the most significant ANNs for this research.

A brief list of the networks analysed in this section follows.

- Reference Neural Network: BatVision Net
- Audio Neural Networks, mainly SoundNet
- Audio – Video Neural Networks:
  - Early Convergent AE Net
  - Two Independent Audio - Video Branches Net
  - U-Net

The results presented are gotten at the same conditions and applying the following parameters and training and validation datasets. After many trials, the values of learning rate and batch size that work better are respectively 0,00005 and 32. The dataset chosen is *Dataset 2*. The datasets for training and validation of all the mentioned nets are composed of apartment data, taken from the entire *Dataset 2*. The validation is composed of some samples from rooms, not comprehended in the training dataset, and also some data from two same rooms considered in the training but acquired in different directions and furniture positions. This configuration of both datasets allows to evaluate if the use of more familiar data, or in other words, data acquired in the same room but in different directions and with diverse furniture positions, during the training phase can improve the performance during the test. It permits to also assess the way in which the dataset affects results and learning efficacy, and to evaluate the modality to approach the dataset expansion.

3.6.3.1 Reference Neural Network: BatVision Net

This paragraph is dedicated to showing the performance of BatVision Net applying the mentioned conditions. The results. shown below, are the reconstructions produced during the validation phase. They confirm the work presented in [10] is replicable and ensure that the acquired dataset is appropriate for this task.

The grayscale result can be compared to the label, posed below. The second column represents reconstruction and ground truth in jet colormap, which is the colormap typology also applied by the Kinect system. The introduction of the results in both the colormap versions is to present a neutral colormap reconstruction and, at the same time, a colormap that highlights some features less evident in the grayscale depth image.

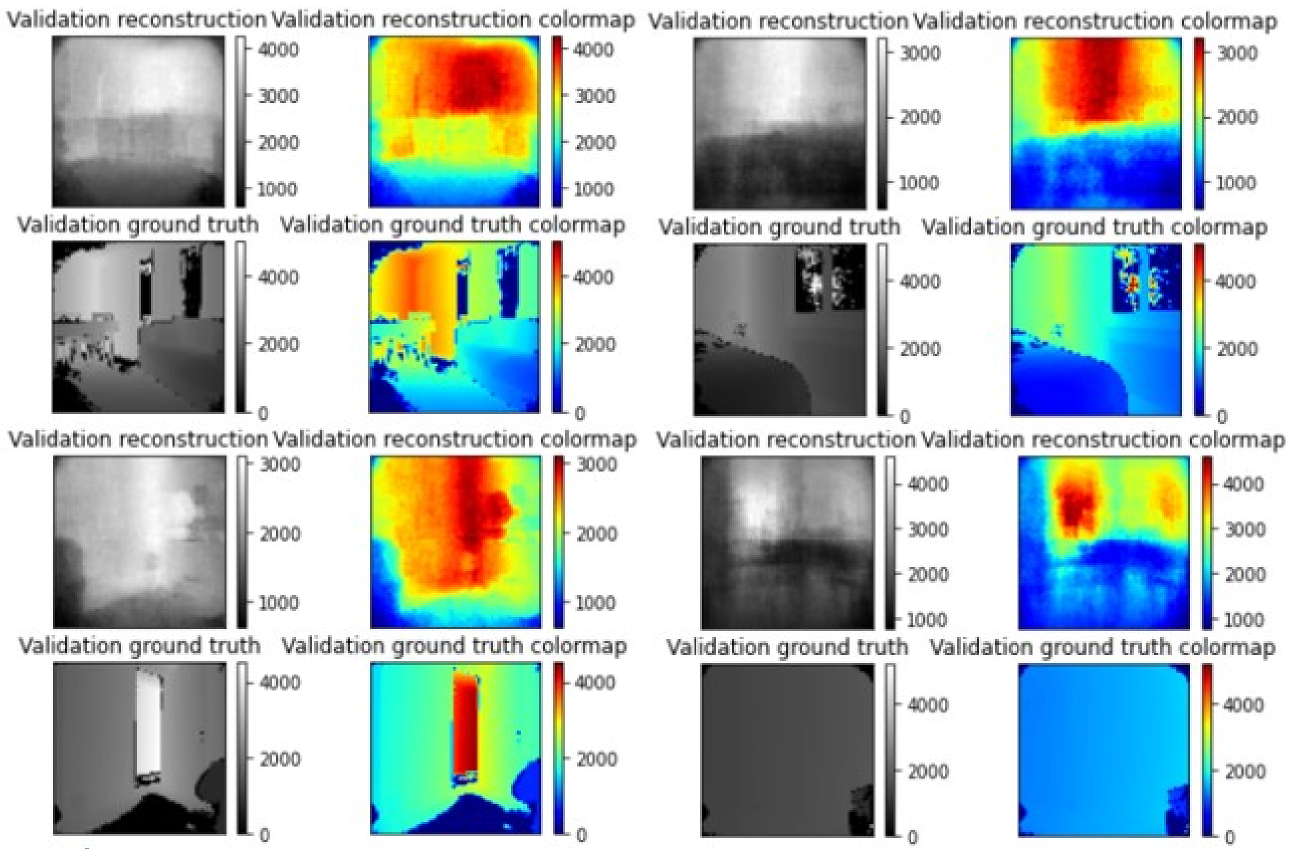


Figure 31. BatVision Net validation results of familiar input data compared to the training dataset

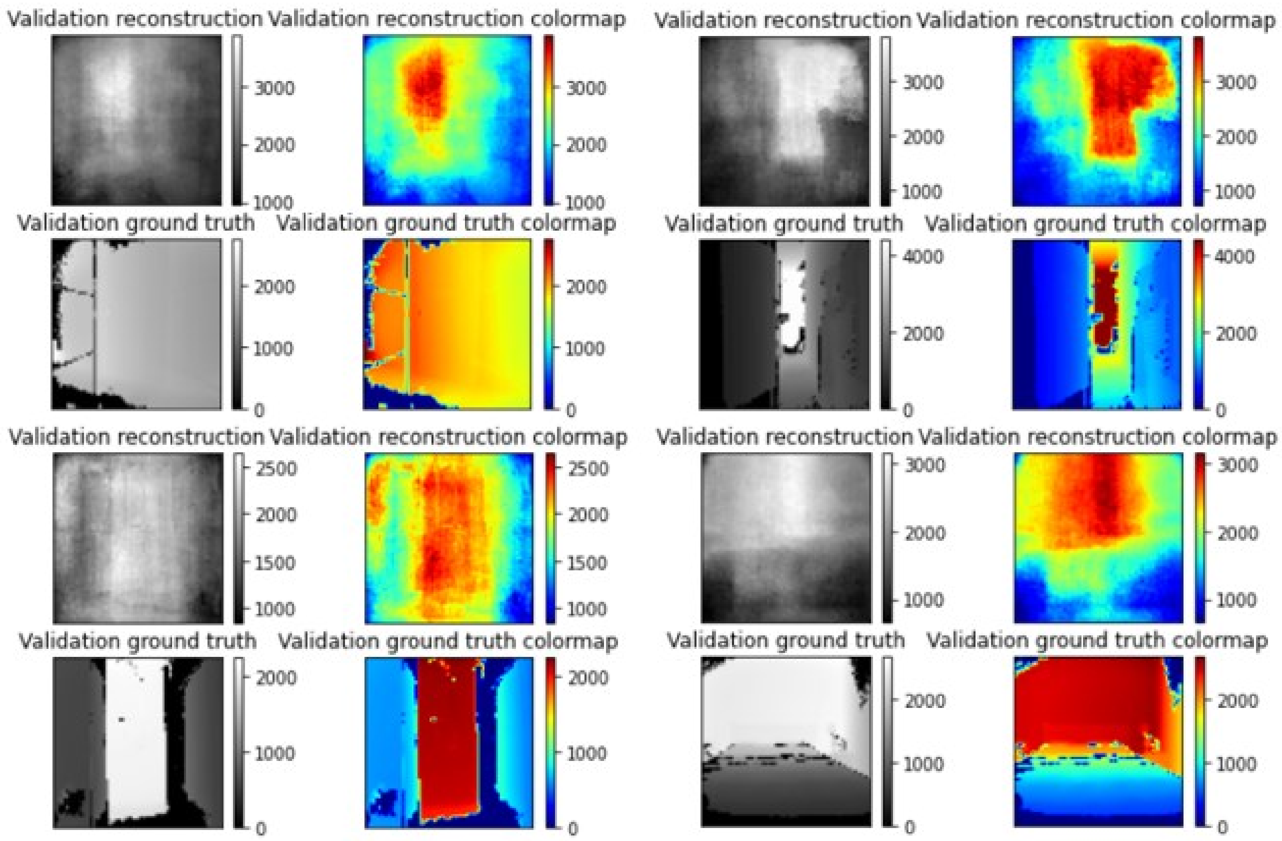


Figure 32. BatVision Net validation results of completely new input data

The training loss trend is shown in the following picture.

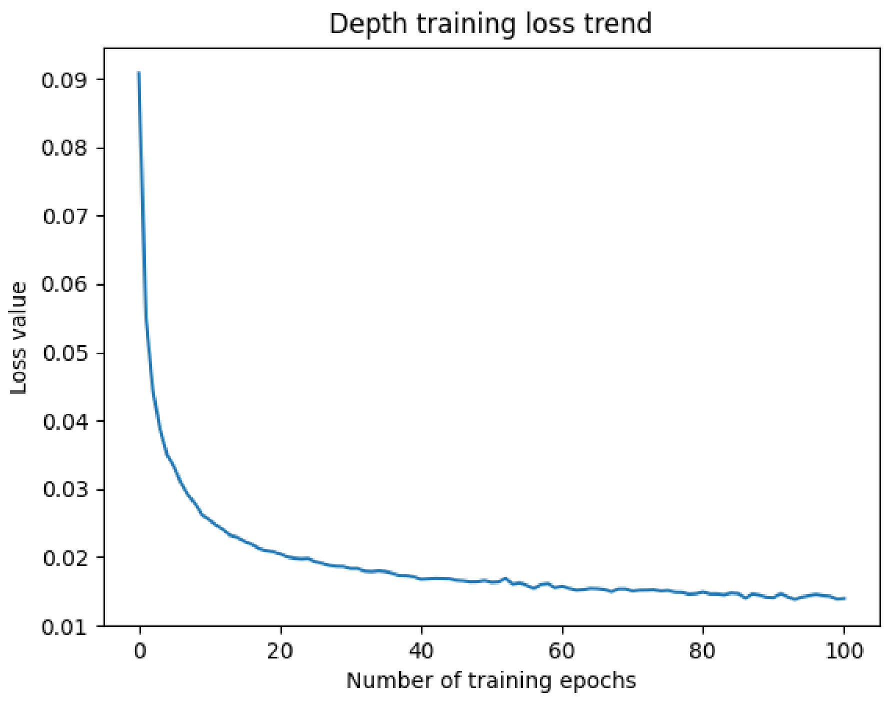


Figure 33. Depth training loss trend of the BatVision neural network, considering a learning rate of 0.0005, a batch size of 32, absolute error as the loss criterion

The training loss is continuously decreasing; however, it does not correspond to a reduction in the average of the absolute error between validation reconstructions and corresponding ground truth. That can be caused by a net behaviour: the more the training epochs assigned, the stronger gets the recalling of training samples or part of them in the validation depth reconstructions, in which the absolute error increases consistently.

The results can show sharper and more reliable reconstructions compared to [10], considering the overall reconstructions provided in the article, even if the dataset is considerably smaller compared to what the authors implemented (ten thousand). The comparison can be effective only at a qualitative level, since they provided only grayscale reconstructions and loss values. Contrarily to [10], the image processing returns grayscale images and grayscale depth matrixes in *uint16* type, instead of *uint8*. This could be mentioned between the reasons that justify the different performance, surely in union with the diversity of the data collection and scenarios.

Some of the results evidently show the need of more data to allow the net to have a more detailed audio feature recognition and scenario depth prediction. However, the neural network outputs confirm that the net is not consistent and reliable.

In conclusion, the only quantitative comparison with [10] that can be conducted is the loss value comparison. The loss criterion chosen in the algorithm presented in [10] is the L1 Loss, that is also applied in for the network of this work. In article [10] the final loss value assessed in relation with their network is 0.0838 for 16x16 images; in relation to the net presented in this work, after one hundred epochs, the loss value corresponds to 0.01396101, performing the reconstruction of 64x64 depth maps, taking into account that a larger dimension of the image means a wider possibility to find errors and failures and a bigger effort applied in training and learning.

3.6.3.2 Audio Neural Network comparison: SoundNet

The conclusions that come from the analysis of the results of different alternative audio encoders are summarized in the following lines.

About the Fully connected Net, the net performs in a worse manner in comparison with the previous, in terms of training velocity and learning efficiency. The conclusions of the fully connected net discussed in

Paragraph 3.6.2.1 are valid also for this type of implementation. where the net achieves the overfitting and reproposes the same reconstruction for each group of inputs. The LSTM Net with sixty-four layers provide reconstructions that are basically composed of a close floor and a central distant spot, to achieve an absolute error in average around five metres for each reconstruction. This result is proposed with any depth image while keeping approximately the same error. The issues are supposed to be in the number of layers and in the complexity apported in extracting features and processing the audio inputs. The next trial is reducing drastically the number of layers from sixty-four to eight: it turns to overfit even in this case, with some reconstruction reproposed for more than one group of input.

The SoundNet results prove that, at the same number of epochs, the audio encoder training is faster and more efficient compared to the BatVision Net performance. It can be appreciated in the following graphs: the first one represents the comparison of the depth training loss trends, the second graph the comparison between the two trends of the average of the absolute errors calculated between depth predictions and corresponding ground truths at every batch.

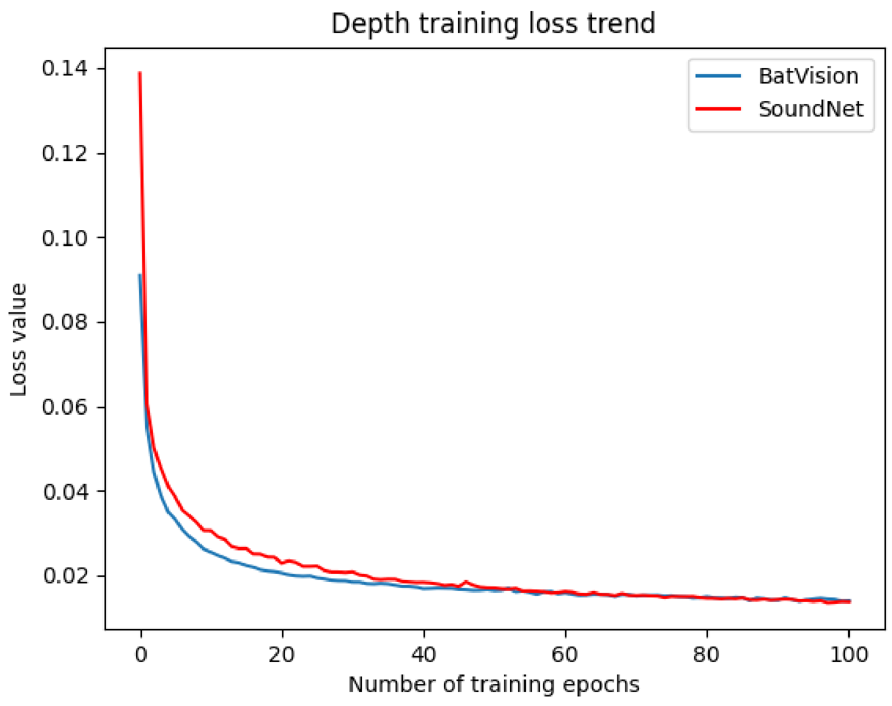


Figure 34. Depth training loss trends of Batvision Net and SoundNet at the same conditions and applying the same number of training epochs.

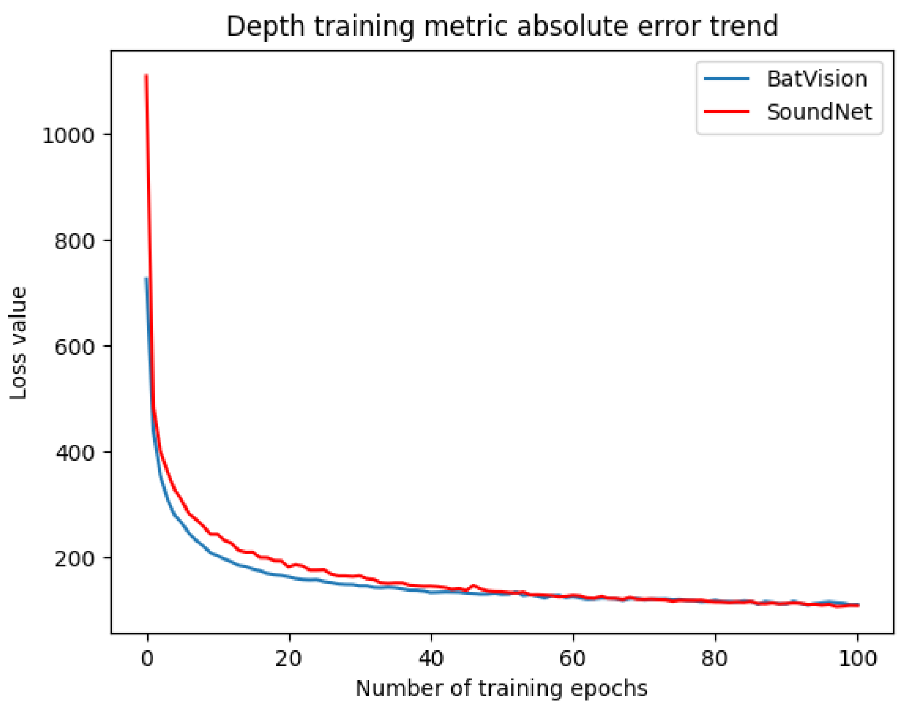


Figure 35. Trends of the average of the depth absolute errors between results and relative labels, from Batvision Net and SoundNet execution at the same conditions and applying the same number of training epochs.

Some of the reconstructions are proposed below. The results are presented in comparison with the corresponding ground truth, positioned below. Both the grayscale and jet colormap are applied, to highlight different feature in the results.

The first set, that can be visualized in *Figure 36*, is composed by the reconstructions obtained during the validation and from the part of the validation dataset part containing the data familiar compared to the training dataset.

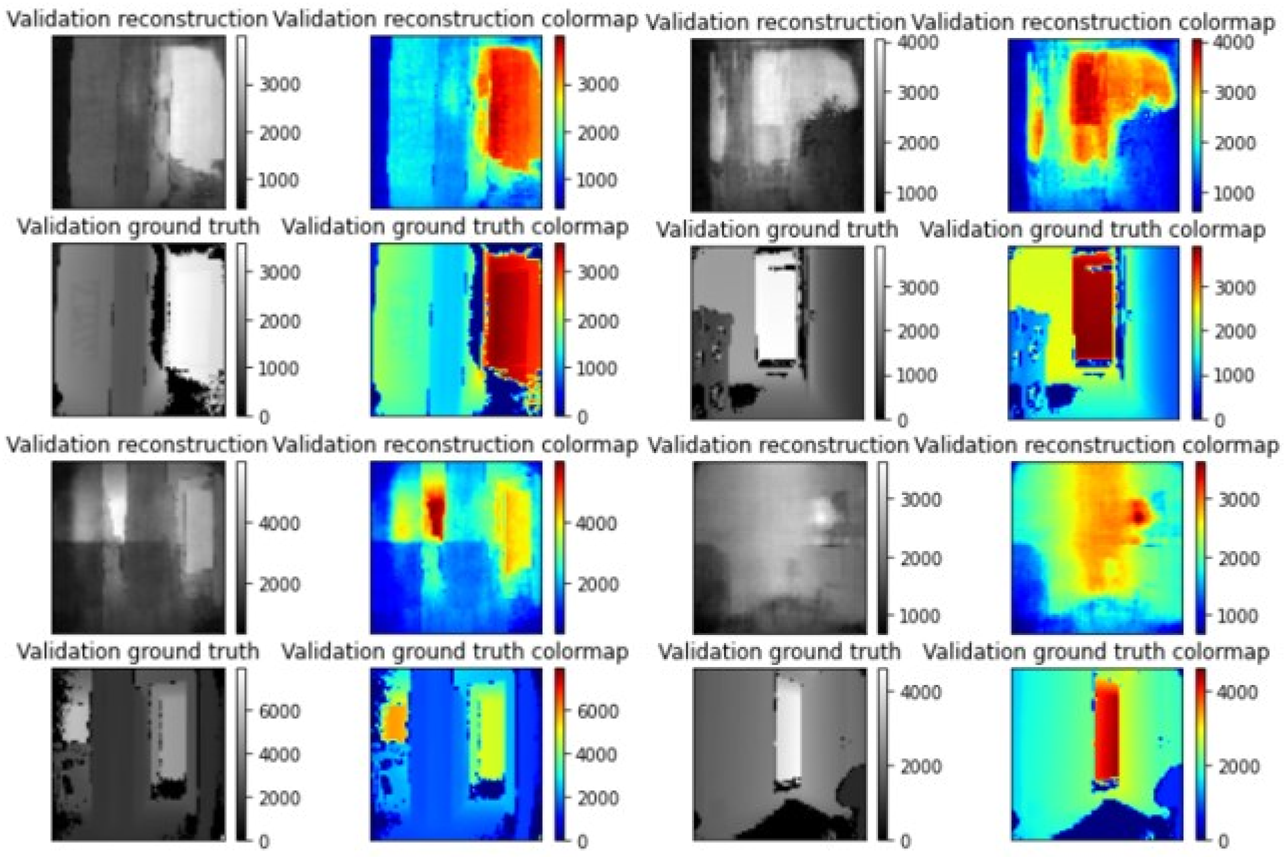


Figure 36. Sound Audio Neural Network validation results of familiar input data compared to the training dataset

The second set, in *Figure 37*, is composed by the reconstruction from the data collected in completely new rooms compared to the ones considered in the training dataset.

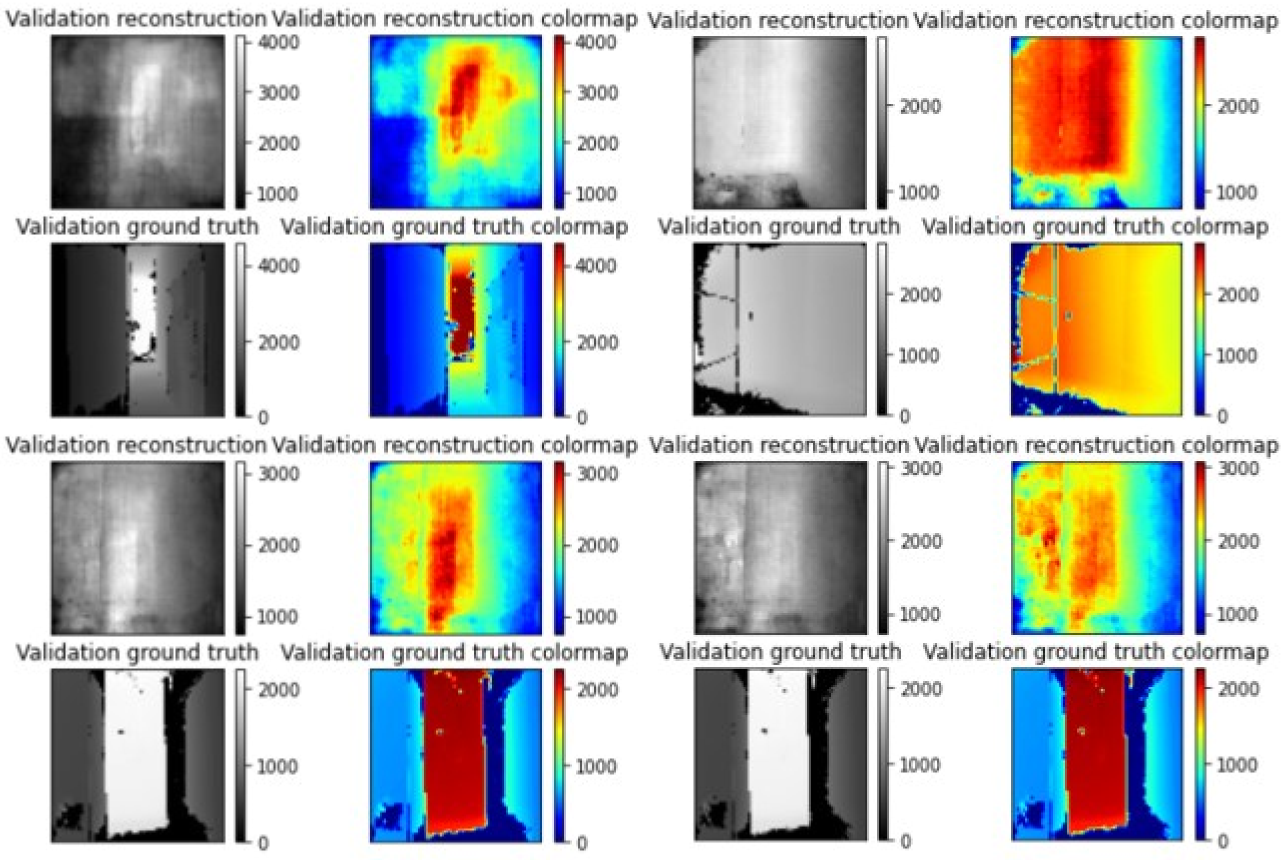


Figure 37. Sound Audio Neural Network validation results of completely new input data

The overall performance demonstrates reliability and effectiveness in the reconstructions, in both the cases presented. The network is faster than the first audio encoder presented in this work, and this facilitates the overall performance and learning efficacy when the encoder is implemented in a wider architecture. The SoundNet version highlights and predicts more features compared to the BatVision version, probably depending on the implementation of MaxPooling layers, matching the expectations.

3.6.3.3 Audio – Video Neural Network comparison

3.6.3.3.1 Early Convergent Audio – Video Autoencoder Net

In the following investigation, the model audio encoder is set as the SoundNet presented in *Paragraph 3.3.1*; all the Image NNs, described in *Paragraph 3.3.2,* are applied and tested as image encoder in this network.

Grayscale and jet results are presented above the respective ground truth version. On the right side, the absolute error between reconstruction and label can be visualized, and the main purpose is to qualitatively evaluate how much the validation reconstruction can resemble shapes and features visible in the corresponding ground truth.

SoundNet and ResNet 18

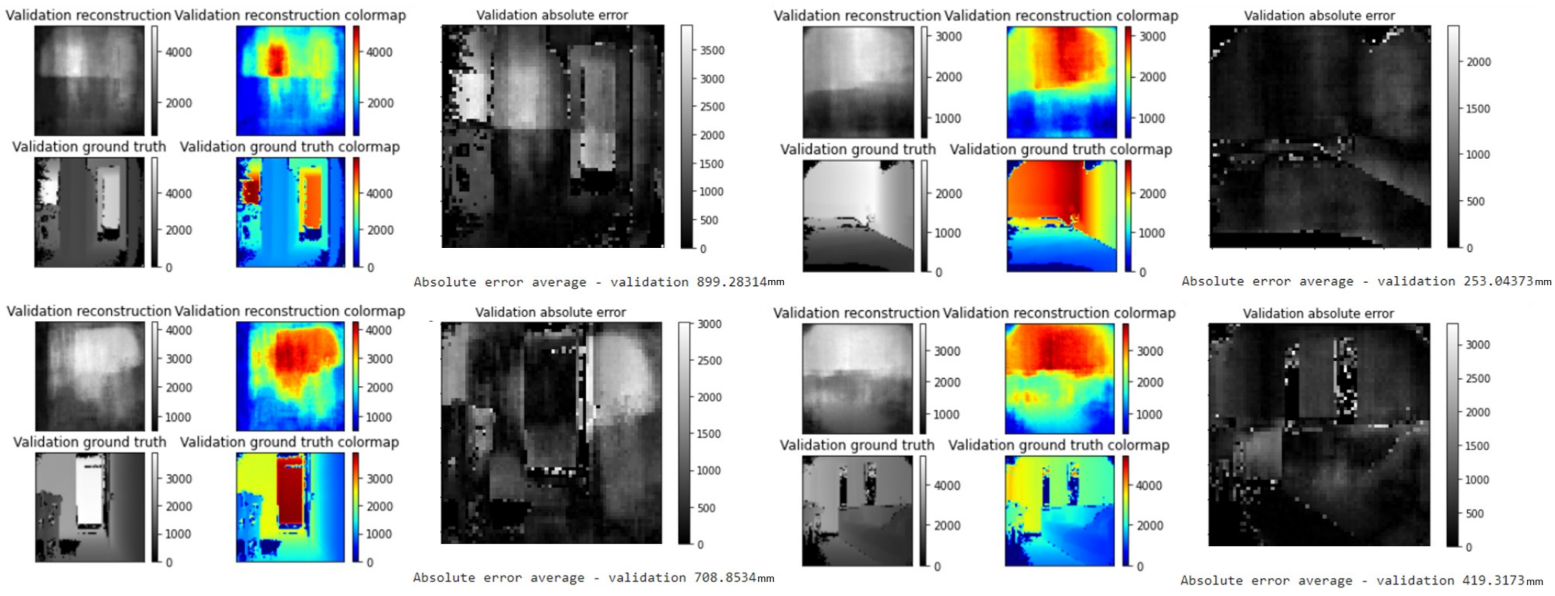


Figure 38. Early Convergent Audio – Video Autoencoder Net validation results from the validation data familiar to the training data. Version: SoundNet and ResNet 18

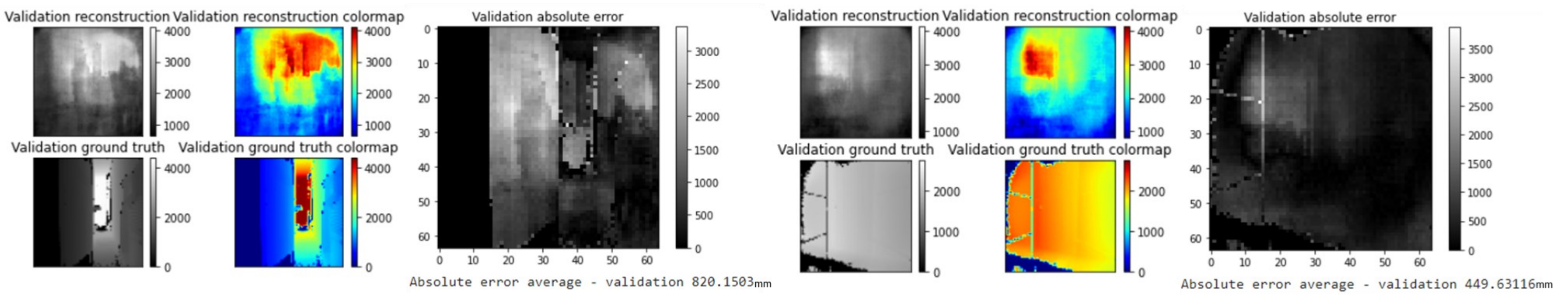
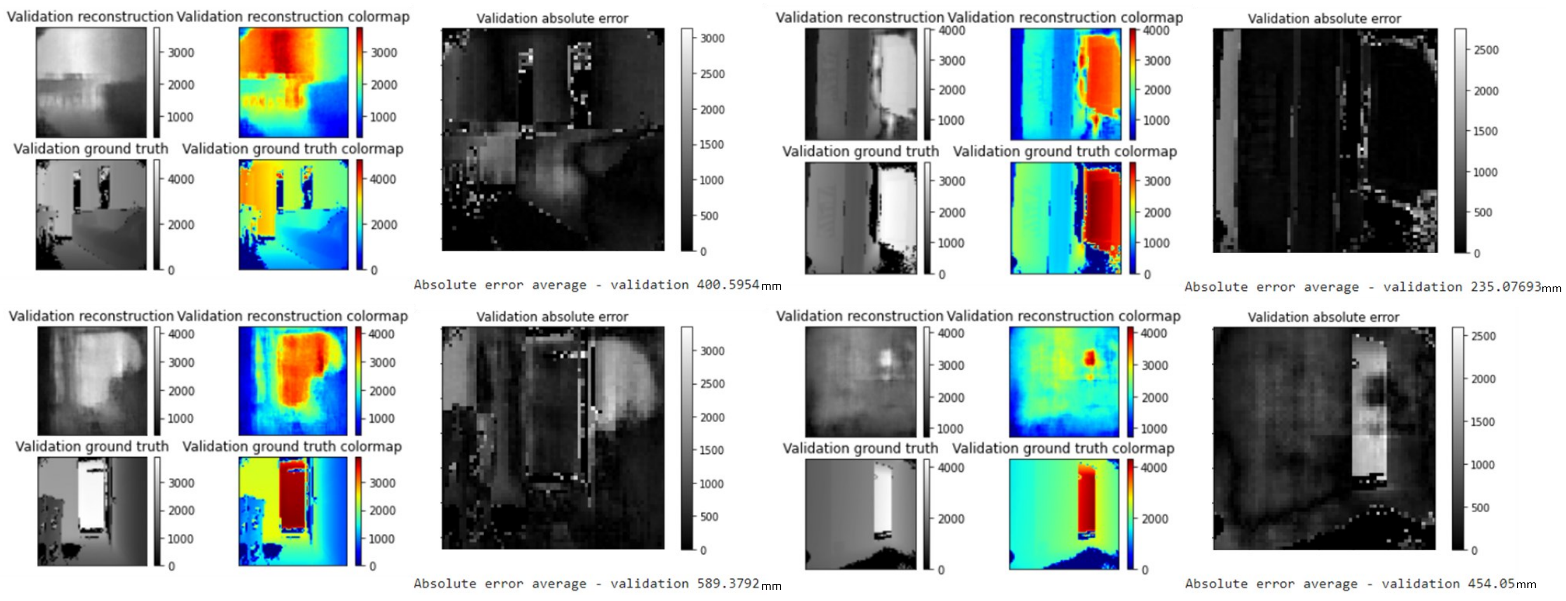


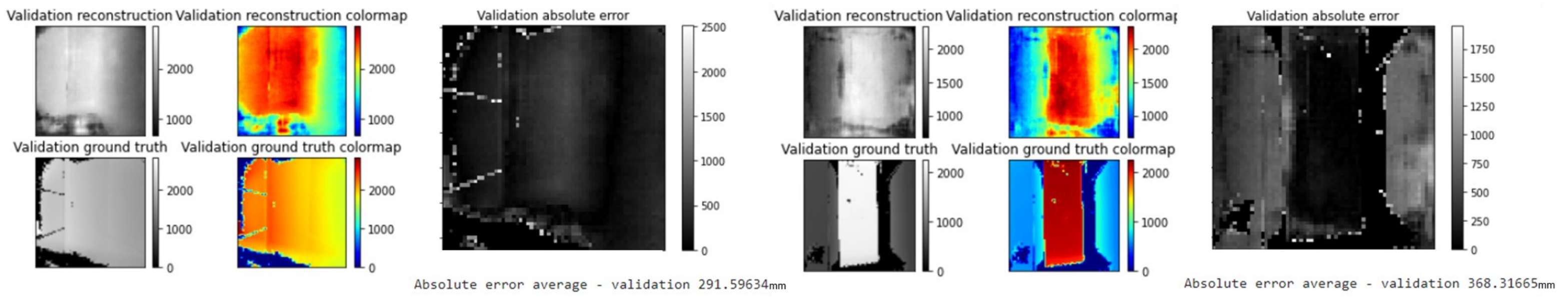
Figure 39. Early Convergent Audio – Video Autoencoder Net validation results from the validation new data. Version: SoundNet and ResNet 18

SoundNet and VGG 16



```
Absolute error average - validation 400.5954mm
Absolute error average - validation 235.07693mm
Absolute error average - validation 589.3792mm
Absolute error average - validation 454.05mm
```

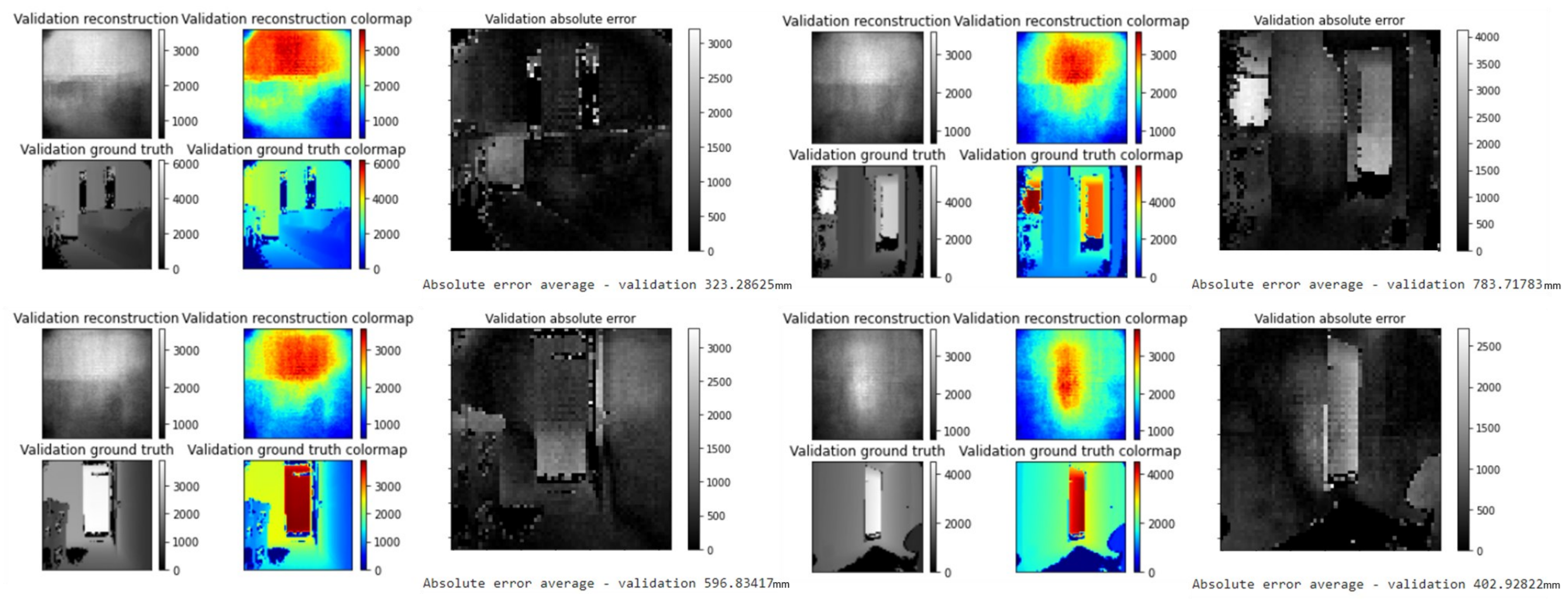
Figure 40. Early Convergent Audio – Video Autoencoder Net validation results from the validation data familiar to the training data. Version: SoundNet and VGG 16



```
Absolute error average - validation 291.59634mm
Absolute error average - validation 368.31665mm
```

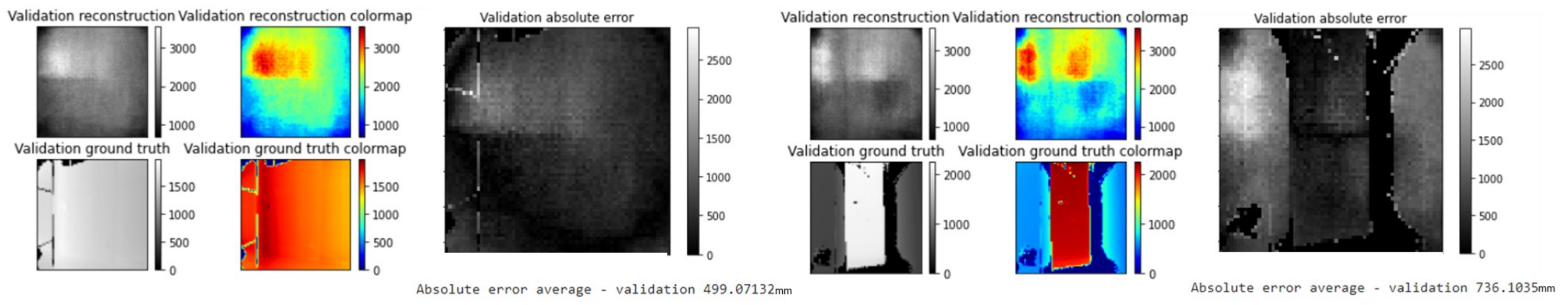
Figure 41. Early Convergent Audio – Video Autoencoder Net validation results from the validation new data. Version: SoundNet and VGG 16

SoundNet and VisualEcho



```
Absolute error average - validation 323.28625mm
Absolute error average - validation 783.71783mm
Absolute error average - validation 596.83417mm
Absolute error average - validation 402.92822mm
```

Figure 42. Early Convergent Audio – Video Autoencoder Net validation results from the validation data familiar to the training data. Version: SoundNet and VisualEcho



```
Absolute error average - validation 499.07132mm
Absolute error average - validation 736.1035mm
```

Figure 43. Early Convergent Audio – Video Autoencoder Net validation results from the validation new data. Version: SoundNet and VisualEcho

The results of the ResNet 18 version show the same behaviour: the reconstructions seem to cut into two different parts, in which the bottom is usually a close floor, and the upper portion is a distant spot. This behaviour can be caused by a not enough large dataset for the ResNet 18 application, but even trying the pre-trained model implementation, the delivered results present the same negative characterization. Another reason could rely on the relation between ResNet 18 image embeddings and the depth decoder processing or also on the poor efficacy of the features extracted by this image encoder for this task. The VGG

16 seems to find consistent image features to reconstruct the shapes of the different scenarios. In some cases, it is possible to recognize the recalling of scenarios seen during the training, however this behaviour could be justified by the need for a wider training dataset. The VisualEcho net results are poor in some cases. In some examples, it can be found the mentioned behaviour of the ResNet 18.

In conclusion, VGG 16 and SoundNet are the most suitable encoders respectively for image and audio feature extraction for this type of ANN implementation; the results are the most reliable compared to the prediction from the other tested options.

3.6.3.3.2 Independent Audio and Video Branches Net

The net output is a depth prediction delivered from echolocation (audio branch) and steered by the monocular RGB reconstruction (image branch), by means of a comparison, in order to correct the depth results by means of the predictions from the image path.

The architecture implemented as the sub-model is the SoundNet NN as audio encoder, followed by the BatVision decoder. For the image descriptor extraction, some experiments are conducted switching from VGG 16 and ResNet 18, with the VGG 16 as the outperformer; the decoder is still the BatVision decoder.

Some results are shown in the following figures.

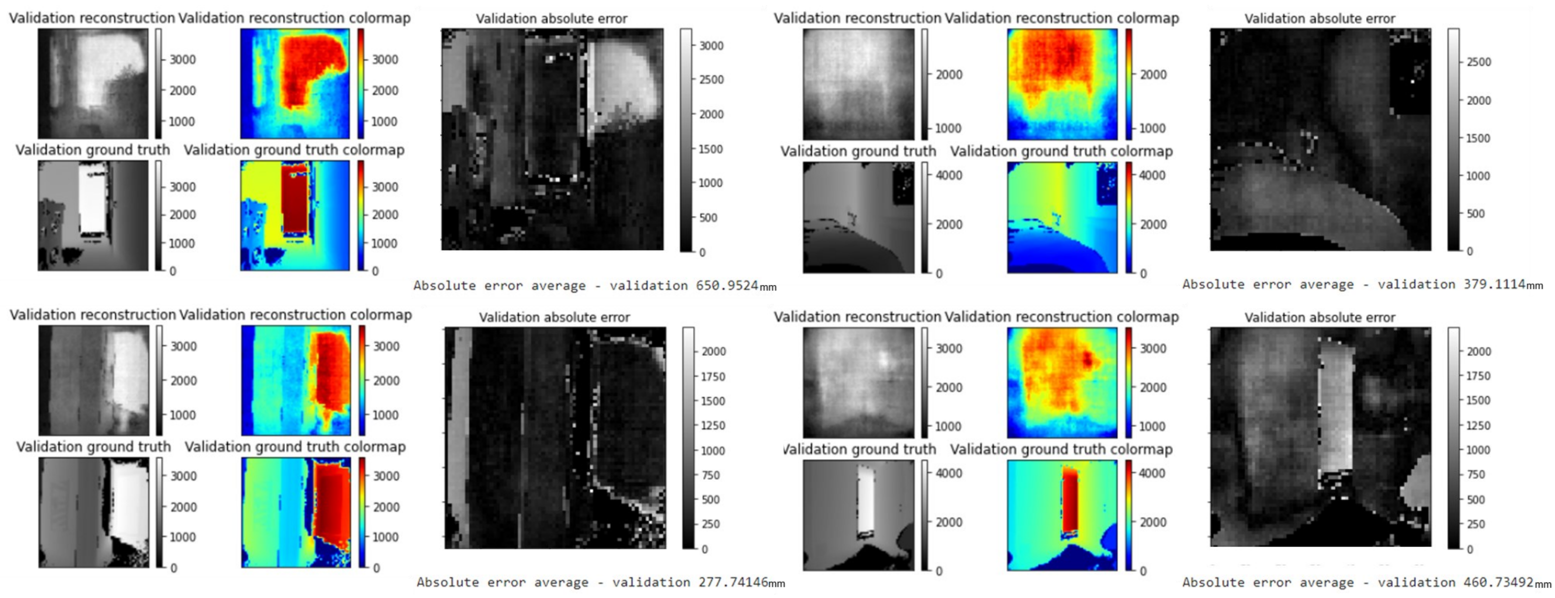
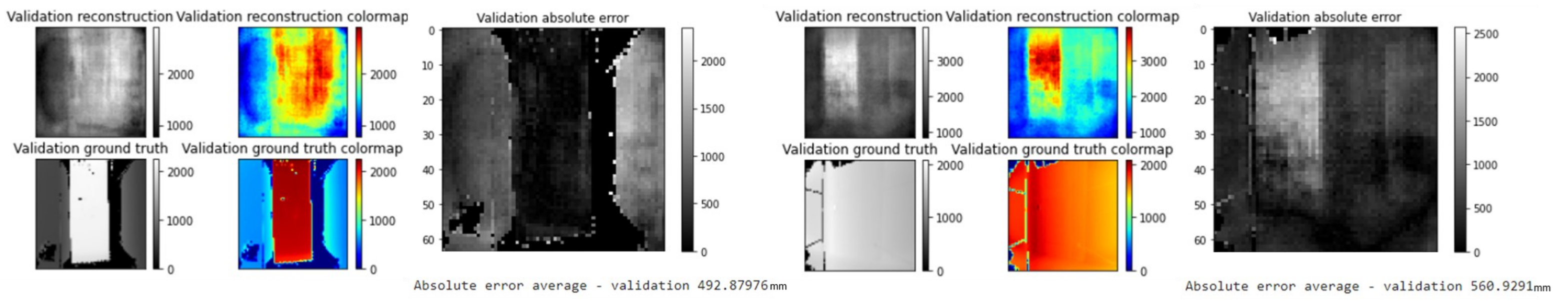


Figure 44. Independent Audio and Video Branches Net validation results from the validation data familiar to the training data



```
Absolute error average - validation 492.87976mm
Absolute error average - validation 560.9291mm
```

Figure 45. Independent Audio and Video Branches Net validation results from the validation new data.

This network outperforms in case of familiar scenarios (*Figure 44*), however in the reconstructions related to the completely new room data the absolute error average is not elevated even if the shapes are partially wrong in comparison with the ground truth. The grayscale matrixes show the distribution of the absolute error: in the first case of the second set, the walls are the issues, as well in the second case the corner is perceived as a distant spot. For these two examples, it is clear that the audio features have a consistent impact compared to the influence of the image branch. Hence, the overall performance matches the expectation: this net presents reconstructions of the depth robustly induced by the recorded reverberations.

The Independent Audio and Audio – Video Branches Net is tested only in the version with same architecture components for the audio branch, the other path is composed by the Early Convergent Audio – Video Autoencoder Net, combining SoundNet as audio encoder and VGG 16 as image encoder. As expected, the results are not considerably different compared to the previous net, however the computational power is higher. The Independent Audio – Video Branches Net is preferable.

3.6.3.3.3 U-Net

Some results are presented in the next figures.

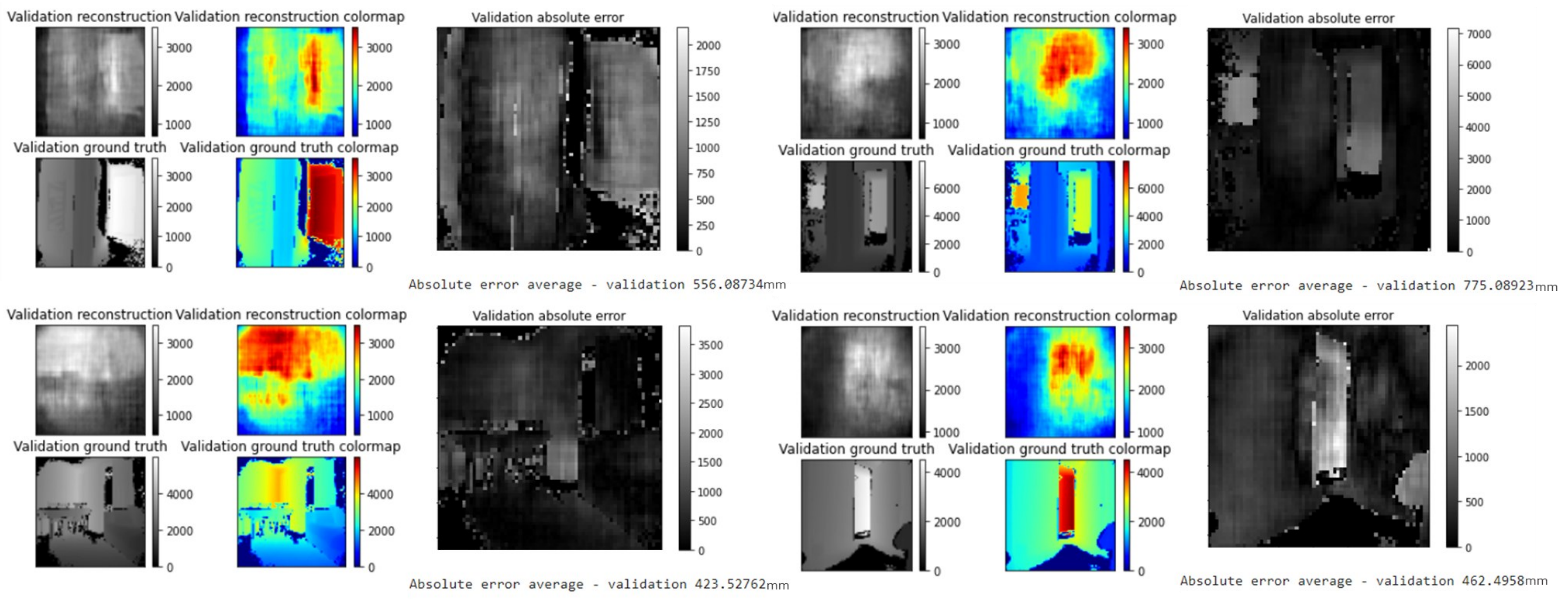


Figure 46. U-Net validation results from the validation data familiar to the training data.

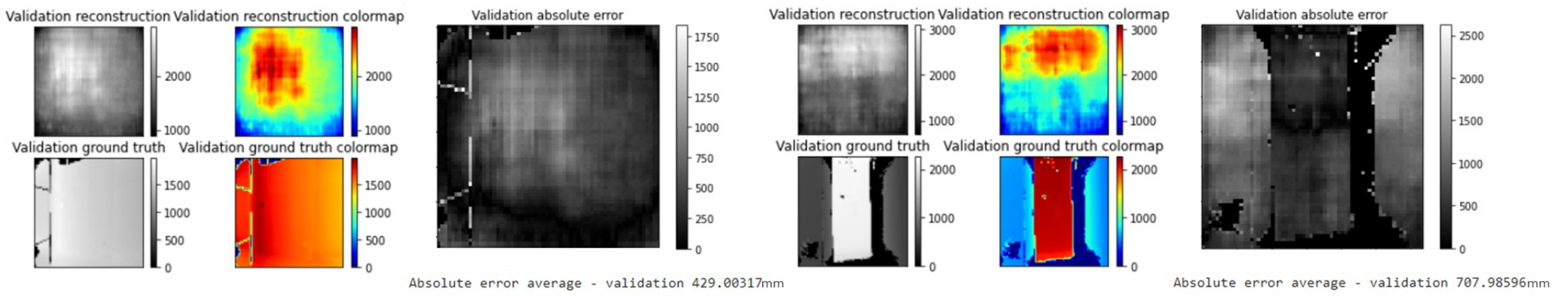


Figure 47. U-Net validation results from the validation new data.

The general behaviour is predicting scenarios consistently but smoothly and not in detail. Overall, the reconstructions present the right positions of the distant spot compared to the ground truth, however these are just presented as spot and not shaped into the actual geometry.

3.7 Last Examination

The last examination is oriented to the Audio-Video Neural Networks that have been resulted the outperformers from the previous investigation. The architectures, chosen from a qualitative and quantitative analysis of the results shown above, considering both the diverse scenario cases, are the Early Convergent Audio – Video Autoencoder Net and the Two Independent Audio - Video Branches Net. The versions selected apply VGG16 as image encoder and SoundNet customized version as audio encoder.

To deepen the examination of the dataset variety affection on the efficacy of the training, the training is conducted through a larger version of the previous training set: the data, acquired at the university meeting rooms and relative kitchens, are added to the apartment set, considered in the previous applications. Moreover, the networks are trained for one thousand epochs to appreciate the effect of a deeper training and to assess the conditions that ensure the best performances. The other parameters are set as in the previous investigation: 0.00005 as the learning rate, batch size fixed at 32 samples. The loss and optimizer criterions are resumed in the following lines. The optimizer criterion is fixed as the ADAM optimizer for both the applications. Regarding the Early Convergent Audio – Video Autoencoder Net, the applied loss criterion is the L1 Loss. The Two Independent Audio - Video Branches Net sees the implementation of the loss criterion L1 Loss, but also of the MSE Loss, in other words, the mean squared error. The depth predictions obtained from the two independent branches are compared to the ground truth and both the losses follow the L1 Loss criterion. The MSE Loss is applied in the comparison between the two reconstructions, to highlight the advantages and disadvantages of both the autoencoder approaches. The nets are trained separately; the tests are conducted in parallel, to compare the reconstructions obtained from the two nets taking the same inputs.

The number of epochs is fixed at three hundreds, as a consequence of evaluating boxplots, averages and medians applied to the medians of the absolute errors calculated over the test batch. After 300 training epochs: the average and the medians for the Early Convergent Audio – Video AE Net are respectively 373 mm and 375 mm, for the second net 382 mm and 426 mm. After 400 epochs of training: for the first net, 470 mm and 455 mm; the second net presented an average of 421 mm and a median of 468 mm. After 800 epochs of training: 498 mm as the average and 475 mm as the median of the first net; for the second net, 423 mm and 430 mm.

The following figures exemplify the typical results of the two nets in comparison with each other and with the corresponding ground truth. Additionally, a figure representing the boxplot is added to every comparison between the test results and it illustrates the quantitative representation of the net performance for the relative inputs.

Boxplots help in data distribution visualization, but also in summarizing significant indexes. As shown in the *Figure 48*, the middle line represents the median (Q2/50th percentile), the first quartile (Q1/25th percentile) is the line before the first half block, the third quartile (Q3/75th percentile) corresponds to the line that closes the next half block. The circles represent the outliers. The maximum value can be found applying this formula: Q3 + 1.5*IQR. If it is used as a subtraction, the result is the minimum.

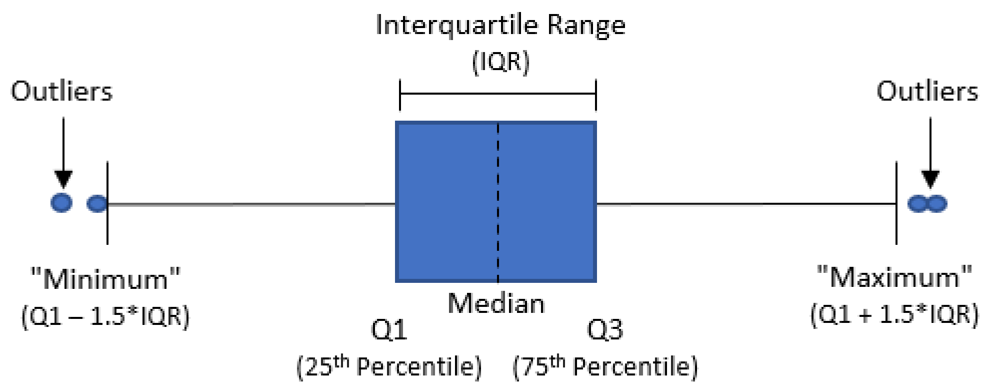
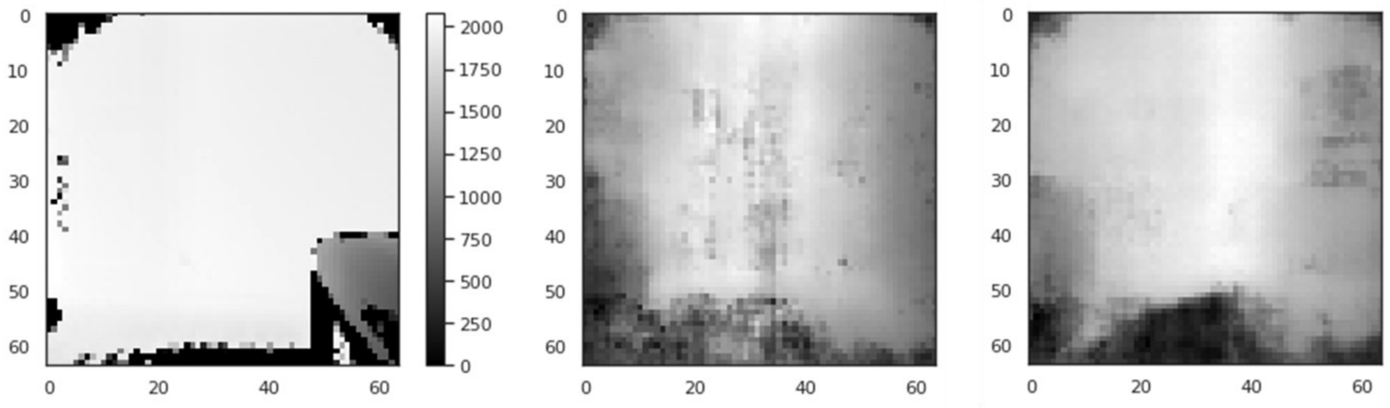


Figure 48. Boxplot explanation figure.

Since the validation dataset is composed by unseen scenarios from some rooms partially seen during the training, but also by unseen scenarios from completely new rooms, the results are divided in order to separate the reconstructions from category to category.

3.7.1 Validation results: unseen scenarios from rooms chosen for the training

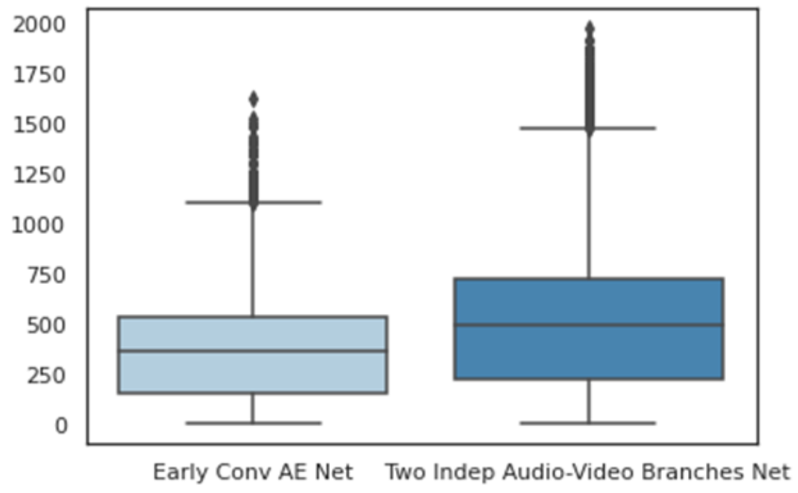
Ground truth | **Early convergent AE Net** | **Two independent audio - video branches Net**



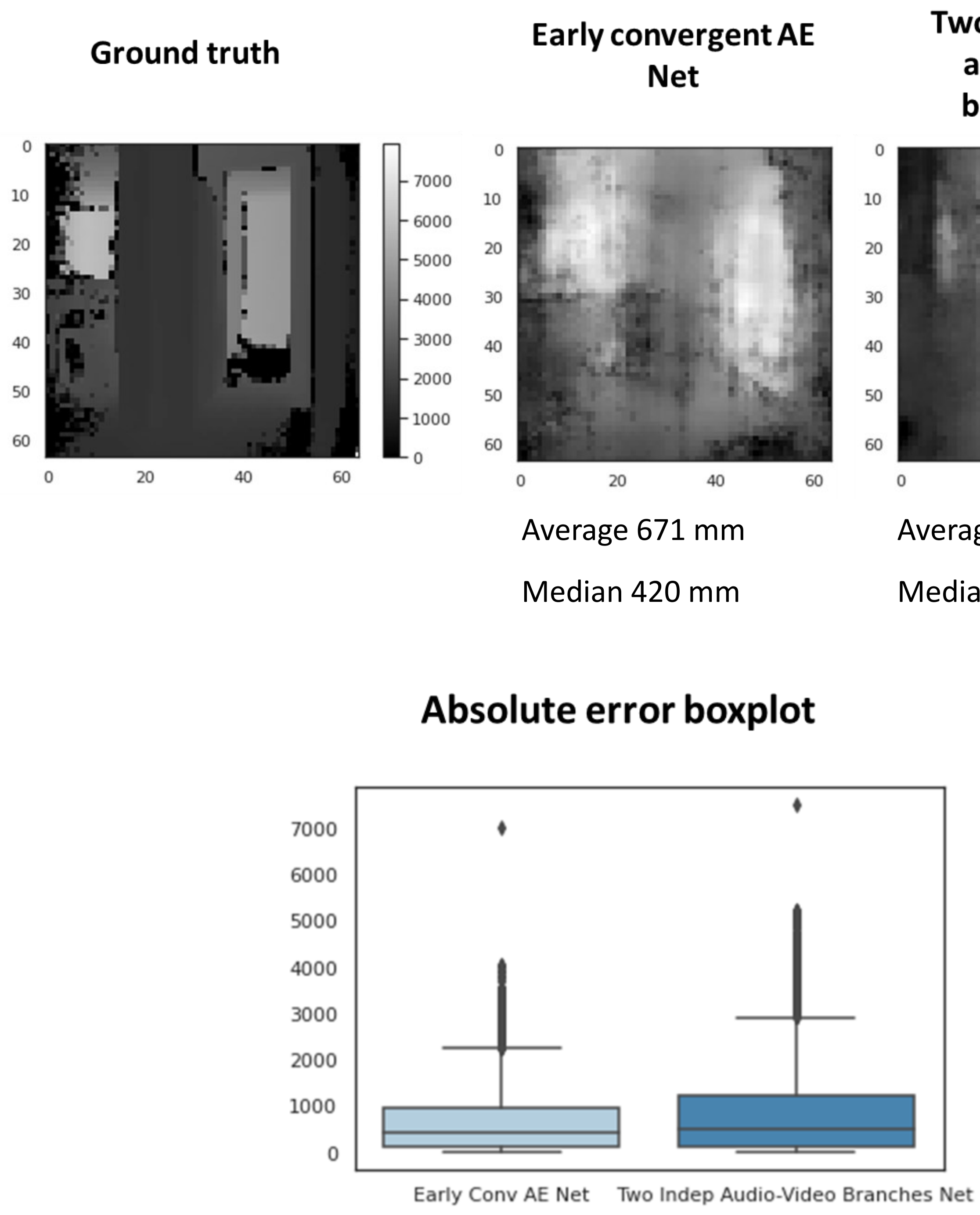
Average 379 mm | Average 514 mm

Median 361 mm | Median 497 mm

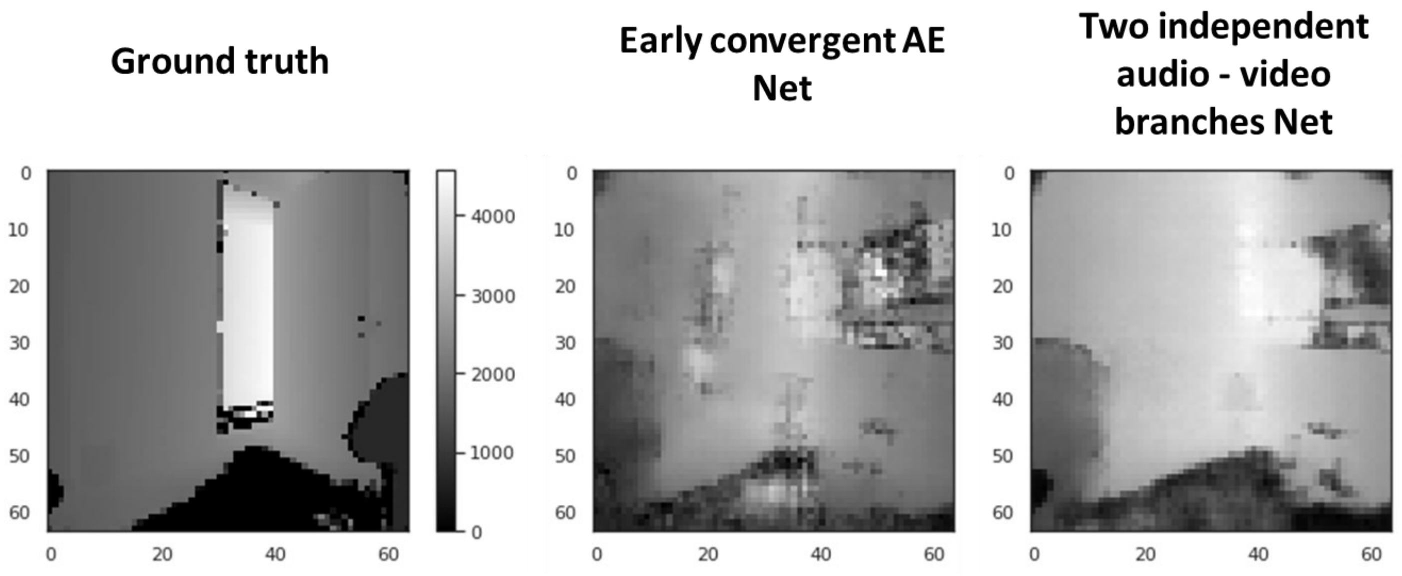
Absolute error boxplot



49 a.


Ground truth
Early convergent AE Net
Two independent audio - video branches Net
Average 671 mm
Median 420 mm
Average 887 mm
Median 524 mm
Absolute error boxplot
Early Conv AE Net
Two Indep Audio-Video Branches Net

49 b.

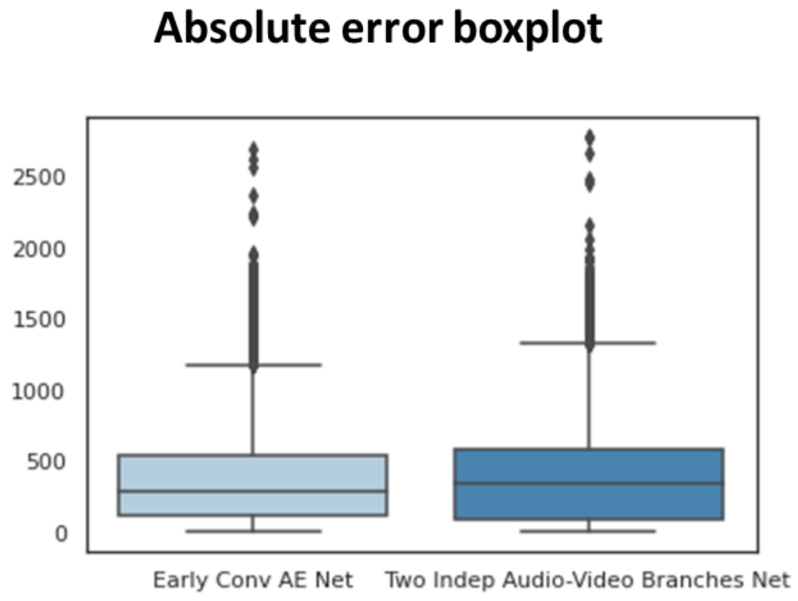


Average 422 mm

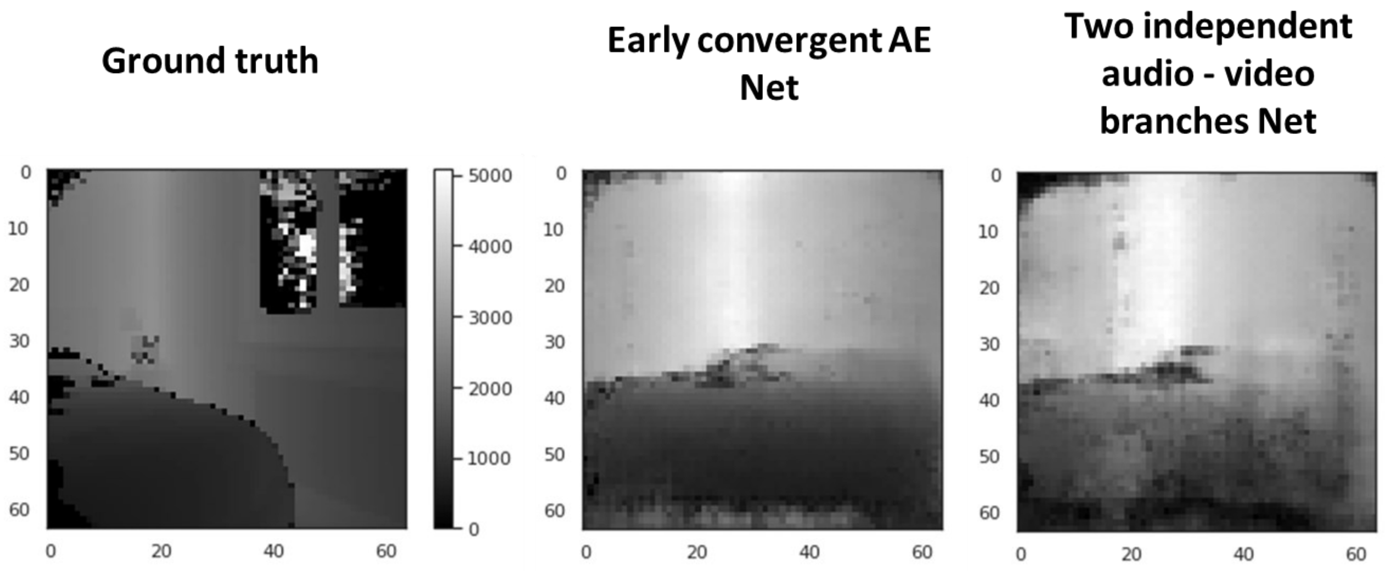
Median 291 mm

Average 463 mm

Median 345 mm



49 c.

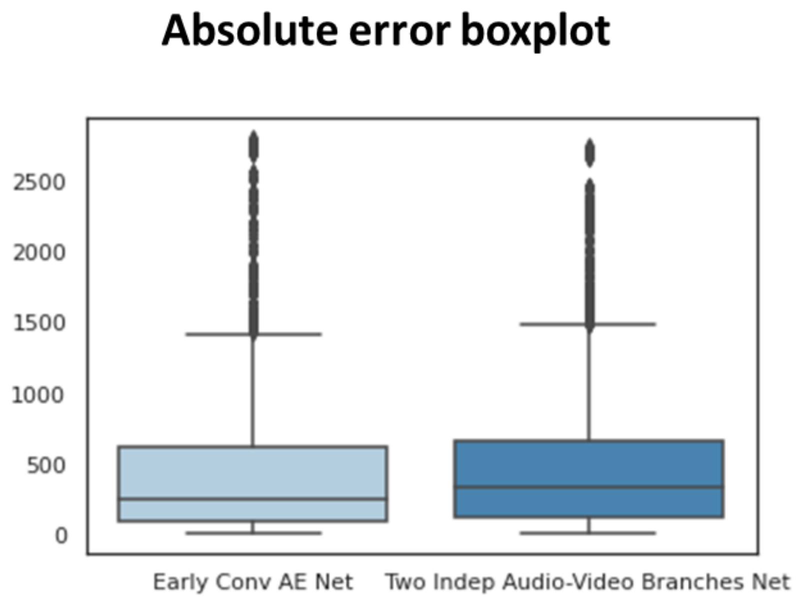


Average 392 mm

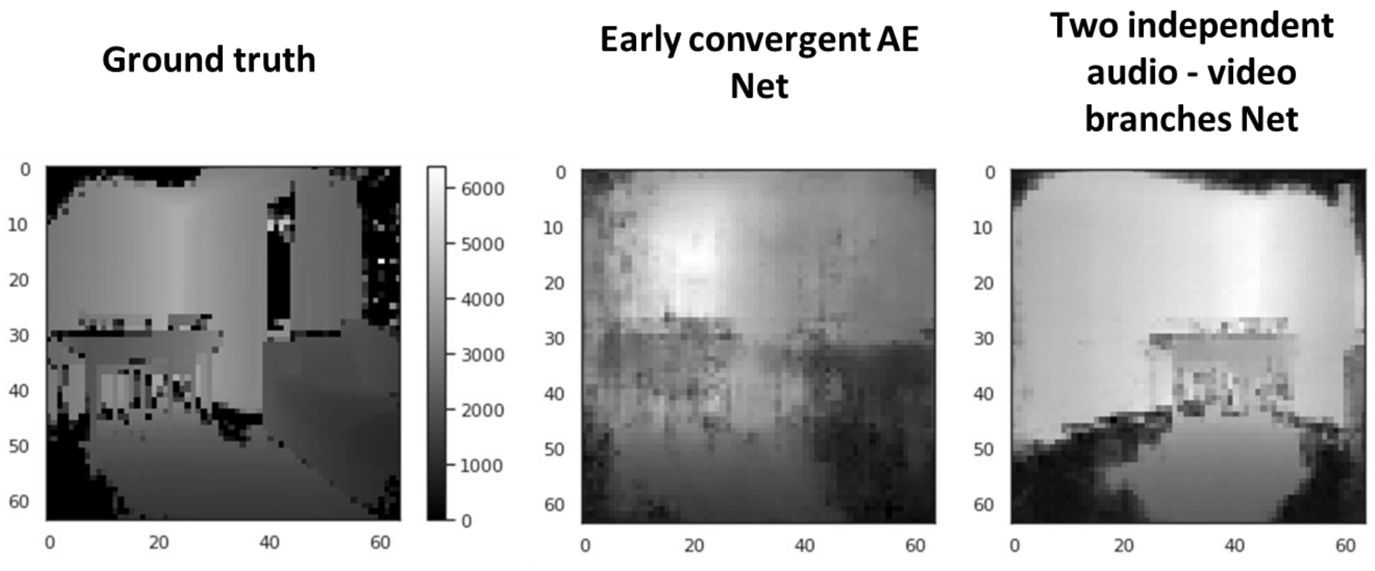
Median 246 mm

Average 439 mm

Median 328 mm



49 d.

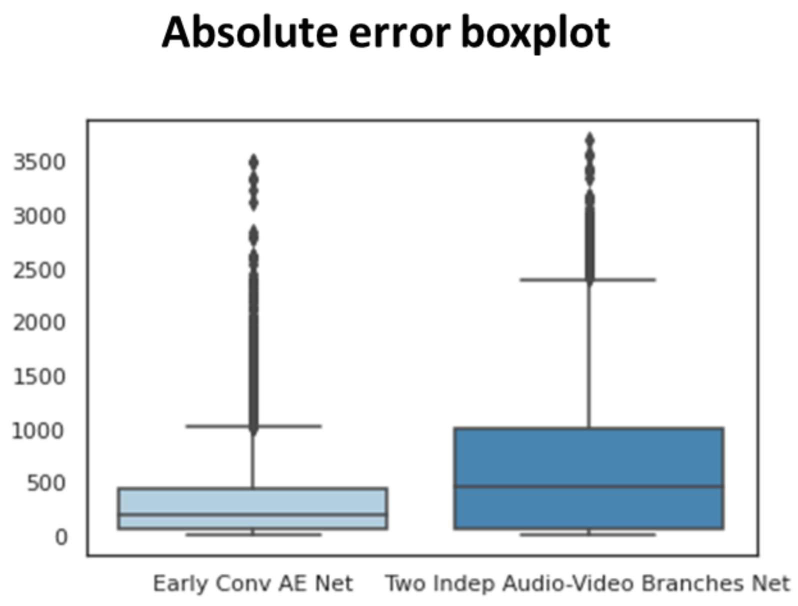


Average 327 mm

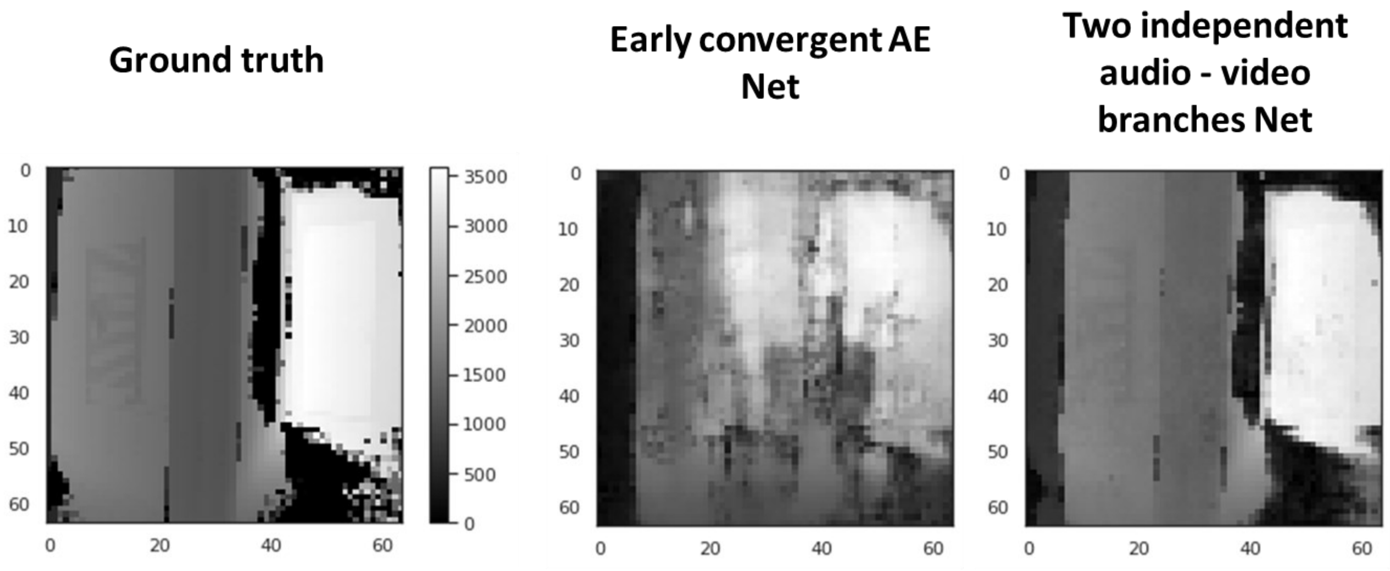
Median 203 mm

Average 642 mm

Median 464 mm



49 e.

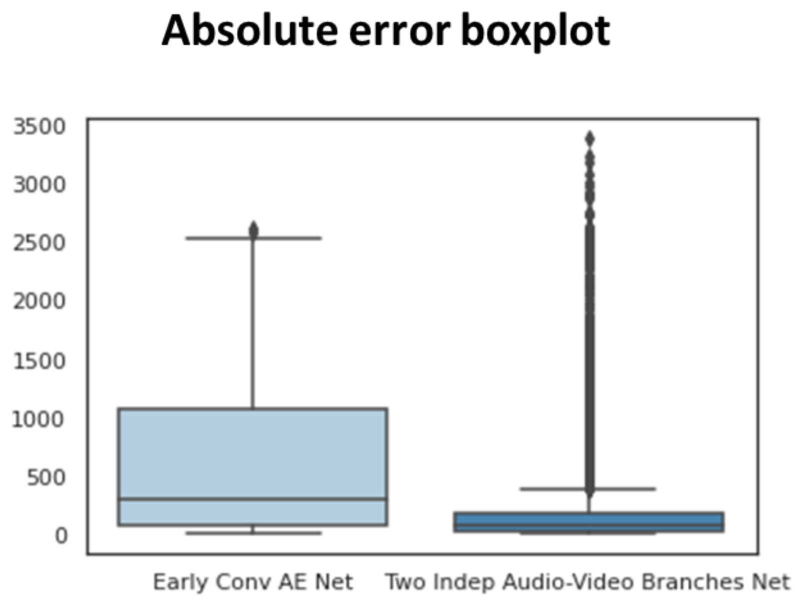


Average 613 mm

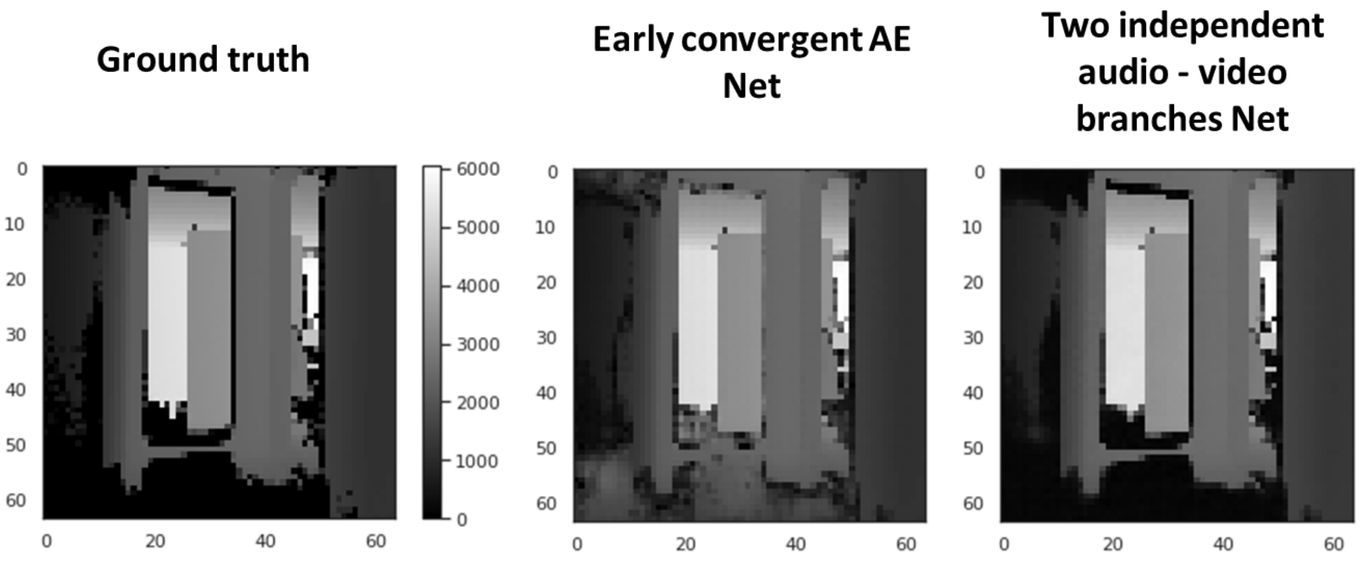
Median 306 mm

Average 270 mm

Median 81 mm



49 f.

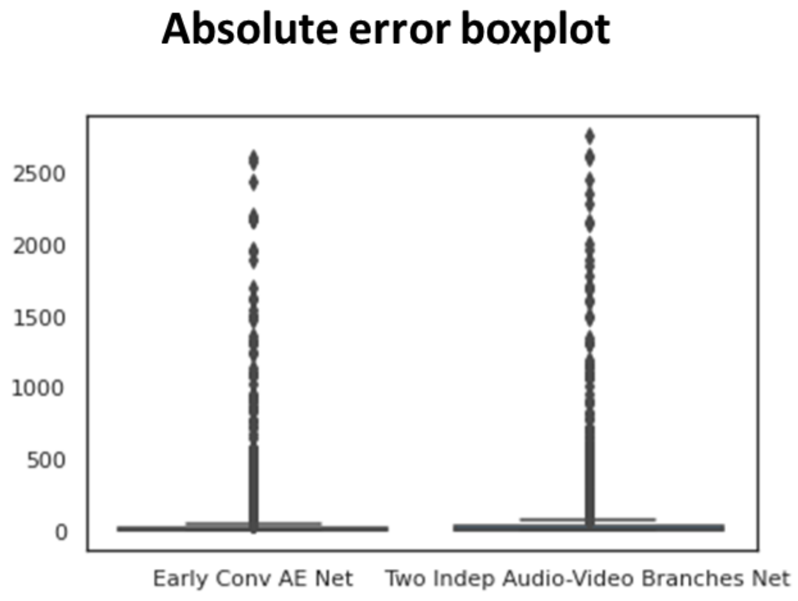


Average 38 mm

Median 8 mm

Average 45 mm

Median 11 mm



49 g.

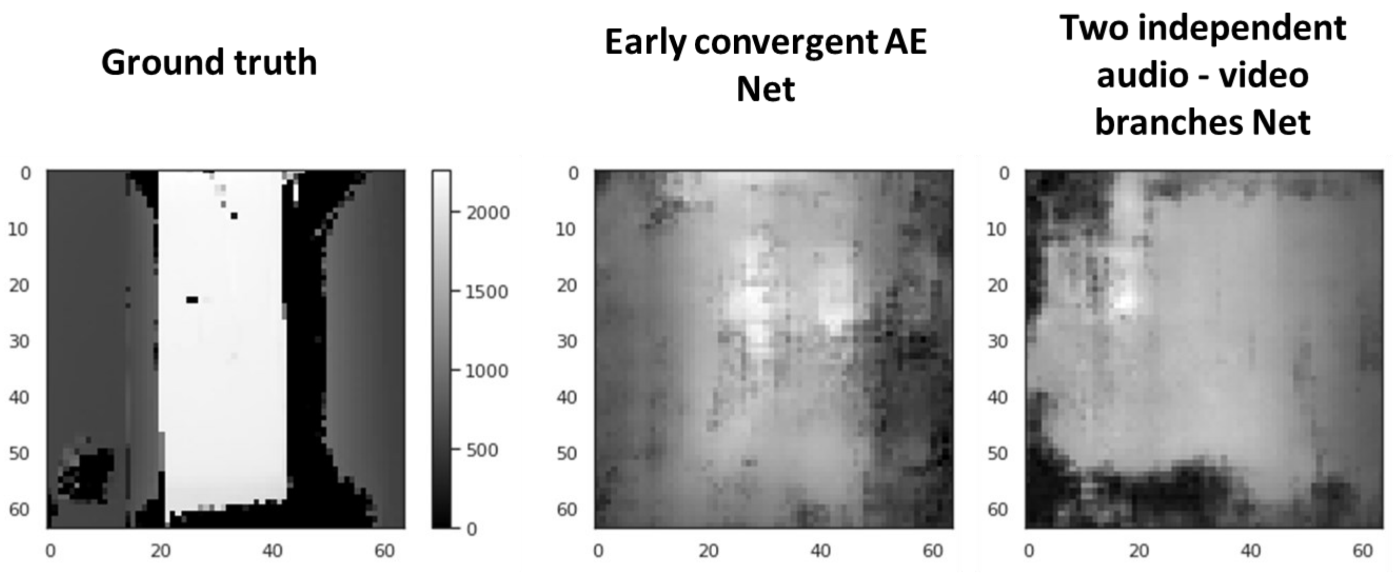
Figure 49. Series of validation results from the validation data familiar to the training data.

From the analysis of these results, it is clear that the two nets behave in two diverse manners.

The general understanding is the Late Convergent Net (the second net, also called Two Independent Audio and Video Branches Net) is consistent with the expectations and comments in *Paragraph 3.6.3.3.2*: the results are robustly influenced by the audio features, however when the predictions from the audio branch are confusing, they are corrected by the image ones. This compensation caused good reconstructions in familiar cases, poor performance in the next results from completely new scenarios. These motivations justify the consistent performance in the examples *49f* and *49e* and the poor performance in cases of challenging sound phenomenon like sound absorption by soft elements (*49d* and 49*e*) and sound dispersive reverberations (*49b*).

The Early Convergent AE Net works with data fusion: the embeddings are concatenated before going through the depth decoder. This approach let the net generalize more and be more independent from what it saw in the training phase. This net outperforms in case of sound challenging scenarios compared to the previous one, like in *49b*, *49d* and *49e*.

3.7.2 Validation results: unseen scenarios from new rooms

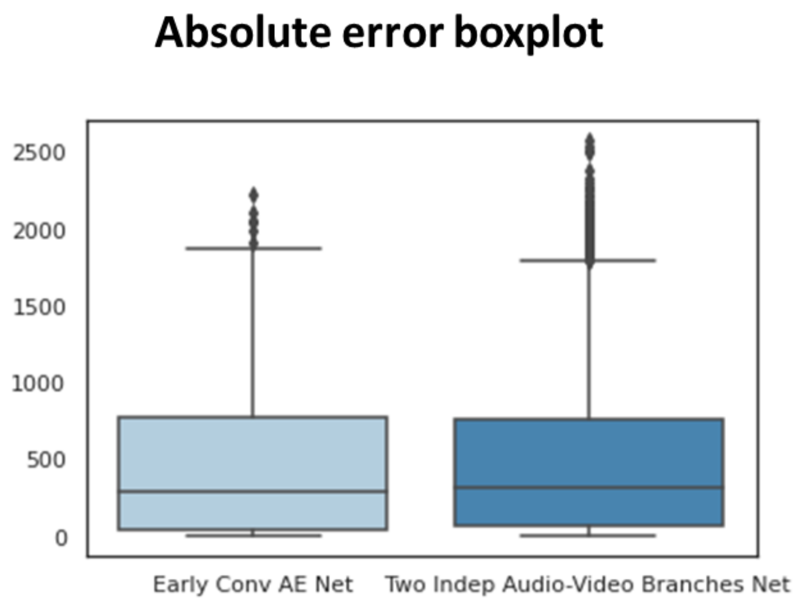


Average 436 mm

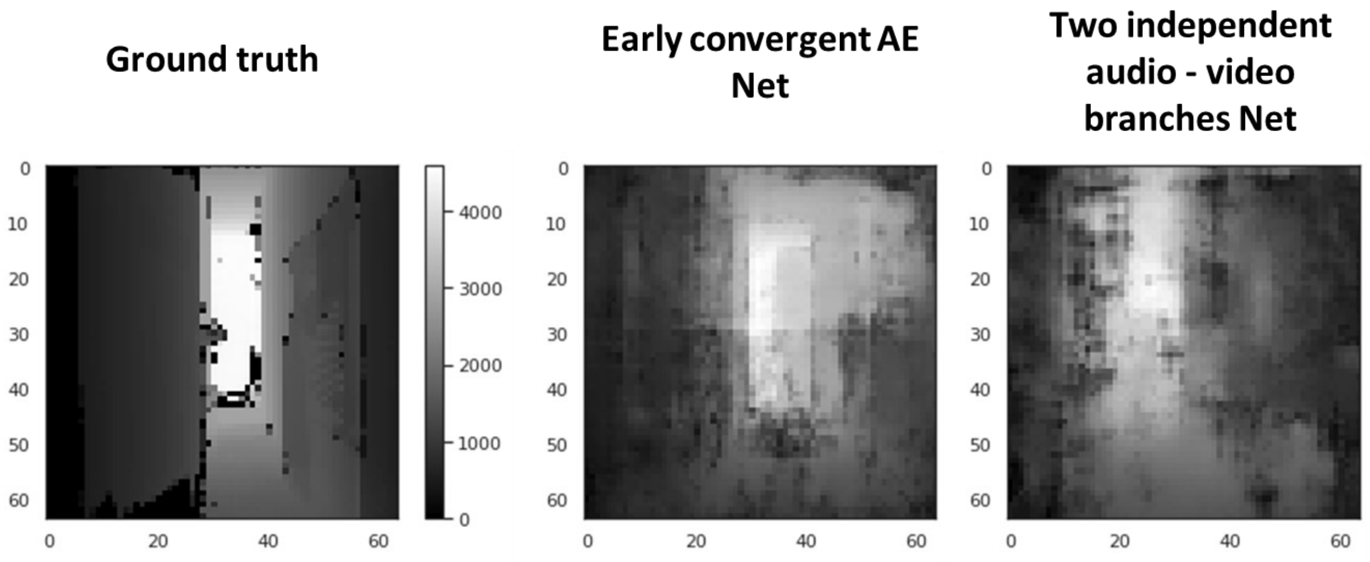
Median 293 mm

Average 541 mm

Median 317 mm



50 a.

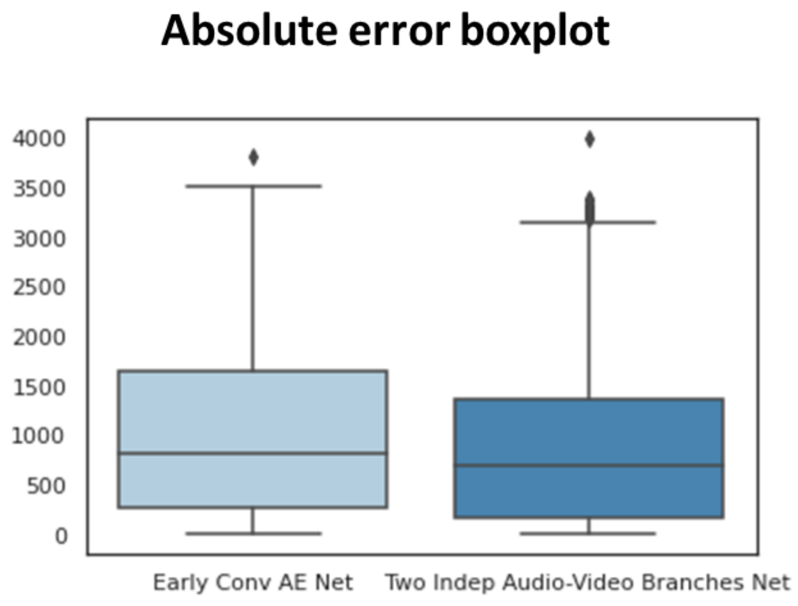


Average 725 mm

Median 566 mm

Average 775 mm

Median 443 mm



50 b.

Figure 50. Validation result from the validation new data.

The scenario *a* shown in *Figure 50*, that corresponds to a shower, is not only an opportunity to see how the net can work with completely new environment input, but also it brings a sound issue: the sound reverberations get closer to the emitted signal in the recordings and the sound bounces between the walls that are close to each other. In this way, some relevant reflected signals can be lost, and the energy of the reflections are lower than in the other normal room scenarios.

Both networks get confused in their reconstructions; however, the Early Convergent AE Net outperforms because it can generalize more compared to the other net, as mentioned in the evaluation of *Paragraph 3.7.1*.

The scenario 50*b* presents another sound challenge, which is the sound absorption, caused by the presence of a soft wall: in both reconstructions it is possible to recognize the hard wall on the left, contrarily the wall on the right is blurry and absent in some portions. Since the Early Convergent AE Net works on a more general level and does not try to reconstruct scenarios seen in the training and possibly matching with the validation case, as the Two Independent Audio-Video Branches Net, it gets the right shapes that are present in the ground truth. Even if at a quantitative level the performances are comparable, the Early Convergent AE Net also outperforms in this case.

The results obtained by means of the application of a completely new room geometry are less sharp and detailed compared to the ones from the previous category. It is clear that both nets need a wider quantity of data to achieve more accurate reconstructions: some of the results recall the training examples; however, matching with the actual corresponding room in case of a scenario seen during the training is generally respected in those cases.

Moreover, the validation reconstructions clearly show the soft material and small room size affections, clearer in the last category examples. The sound absorption and reflected signal distortion caused by a soft wardrobe are visible in the last example, or additionally in reconstructed the sofa distances; the previous one can be considered as an exemplification of how the reverberations can be confusing in a small environment like a shower. In general, the overall performance of the net shows a higher level of accuracy in spaces with rigid and hard walls and with hard and simple objects and furniture.

From a quantitative analysis based on the average and the median calculated over the absolute errors of all the reconstructions in a validation batch, the Early Convergent Autoencoder Net outperforms with this dataset and fixed conditions in comparison to the Two Independent Audio - Video Branches Net. The actual values are shown in *Table 10*.

Early Convergent Autoencoder Net		Two Independent Audio-Video Branches Net	
Average of the validation medians	Median of the validation medians	Average of the validation medians	Median of the validation medians
372,993825	374,7126465	381,7882767	426,3030701

Table 10. Average and median calculated over the absolute errors of all the reconstructions in a validation batch for the Early Convergent Autoencoder Net in comparison to the Two Independent Audio - Video Branches Net

The table show that in both cases, the median is lower than the average: there are some outliers that drop down the average, hence the comparison from a numerical point of view should focus on the medians.
In conclusion, the Early Convergent Autoencoder Net can generalize more compared to the Two Independent Audio - Video Branches Net, which, contrarily, outperforms in finding the similarities between training and test datasets. Hence, the second neural network outperforms when applied to a reduced number of scenario cases, proposed in a large dataset; differently, the other one works better in relation with a bigger variety of scenarios and room architectures provided through a larger dataset.
However, both the analysed networks outperform, not only at a qualitative level, but also at a quantitative level in comparison with the other nets. The following table proposes average and median values, calculated over validation reconstructions of a batch for the two networks deepened in this chapter, but also extends the comparison to the BatVision customized version presented in this work. The values are delivered considering the training and test performed with the mentioned data distribution in training and validation dataset.

Early Convergent Autoencoder Net		Two Ind. Audio-Video Branches Net		BatVision Net	
Average of the validation medians	Median of the validation medians	Average of the validation medians	Median of the validation medians	Average of the validation medians	Median of the validation medians
373 mm	375 mm	382 mm	426 mm	415 mm	431 mm

Table 11. Average and median calculated over the absolute errors of all the reconstructions in a validation batch for the BatVision Net in comparison to the Two Independent Audio - Video Branches Net and the Early Convergent Autoencoder Net

4 Conclusion

The purpose of this thesis is to prove that it is possible to get reasonable depth predictions even in small indoor spaces, with affordable sensors, without the use of silicon ears to capture the sound, providing audio raw data in the audible band as input of neural networks.

The contributions of this work are the collection of a sharable dataset of corresponding audio recordings, images, and depth samples in many diverse indoor environments; the application of real-world data as inputs of the neural network; the implementation and test of multiple architectures and following assessments, evaluation and comparison between the audio depth prediction and the audio-vision version; unneeded bat-shaped silicon ears to estimate the depth through the echolocation. Moreover, all the networks in this work deliver depth maps in metric values as results.

The limits of working with sounds in the audible range are diverse, starting from the noise in public or private indoor environments. Moreover, the choice of rooms, with a maximum measurable distance lower than eight meters, introduces some issues, like dealing with the reverberation phenomenon and the superposition of emitted and reflected signals. Also, the use of real-world data does not allow to benefit of the advantages of simulating the echo: synthetic audios can contain clear reflected signal recordings, not including the superposition of the emitted sound and noise. Even with the mentioned issues, the results clearly show that is possible to obtain valuable depth maps in both the modalities of audio implementation to emulate the echolocation and multi-sensorial application to imitate the ability of bats to privilege the most significant information from the perceptions of different senses.

The results, produced in other works, are usually delivered with a typical image data type: uint8. However, the Kinect V2 system provides depth arrays in uint16. Hence, the depth data are fed into the nets with the original data type from the Kinect and the network reconstructions are delivered with the same type, in order to achieve a higher accuracy of the network training process and, subsequently, of the depth predictions. Moreover, the results are not just provided as qualitative maps in grayscale; the depth reconstructions are depth arrays in metric values, in which each value associated with a pixel effectively corresponds to the estimated distance between sensors and the obstacle.

The dataset validation allows to demonstrate that the final dataset can be interpreted as an opportune fit for this task and to ensure the efficacy of the next trials and subsequent assessment. Additionally, it guarantees the replicability of what is declared in [10].

The SoundNet customized version, presented in this work for audio feature extraction, has a positive impact on the depth reconstructions from audible data. Thanks to this implementation, tangible improvements have been achieved in the Audio NNs, even if the dataset has reduced dimensions in comparison with the one that was utilized in the [10] dataset (ten thousand data).

The implementation of RGB images surely increased the reliability of the predictions. It confirms the initial expectations: the multi-modality application enhances the performance and handles the uncertainties, as proved in the last chapter. The neural network predictions are more robust. The drawback of an increased request of computational power and memory to store a wider dataset is easily managed thanks to a cloud drive and Google Colab Pro Plus resources.

Another outcome of this work is the presentation of examples that demonstrate that the presence of soft obstacles and walls in indoor environments affects the depth predictions obtained through the application of audio recordings in the networks described in this work, because of the phenomenon of sound absorption. An approach to this issue could take inspiration from what the authors of [11] have proposed for synthetic data: hypothesizing material information from the RGB images and increasing the available information taken as input by the net could develop an improvement of the prediction accuracy in this typology of critical scenarios. The problem might also reduce its impact in response to an increased dataset and deeper training. However, the problem could be also approached considering sound and not only a possible association between sight and a material pretrained classification: the material recognition could be accomplished by analysing the reverberations and training an ANN to learn the association between audio features and materials.

In any case, every architecture demonstrates to need for a larger dataset to enhance the test performances. Despite the increment of the scenario variety in the apartment data supplied in the training phase and even using a modest number of data, increasing the amount of training data, by adding the samples acquired in the university meeting rooms and kitchens, has produced an enhancement in the accuracy of test predictions. Hence, the next step to move forward in this research is surely to increase the number of data and the variety of indoor scenarios. The ideal development could take inspiration from what the Replica and Matterport datasets have accomplished for 3D reconstruction and acquisition of RGB images: a data collection suitable for indoor applications of the presented nets should be conducted in many apartments, to provide a wide range of various room architecture and furniture.

The expansion of the dataset and its subsequent sharing could represent a valuable resource for future research on this subject. It might be interesting to apply the dataset that the authors of [10] will publish in January 2023. Additionally, a possible resource for this purpose could be a Zurich hotel, where the managing director demonstrates willingness in supporting a dataset acquisition. The dataset expansion should be accompanied by an improvement in audio-video synchronization and sample association.

Another suggestion for the subsequent starting point in this research is a deeper investigation of the emitted signal. The Room Impulse Response can be analysed in relation to the voices of bats and humans' mouth clicks. Deepening this research could collaterally help the development of the technology and the devices for supporting the orientation and navigation of blind people and for echolocation training. Surely, it would be interesting to understand and subsequently reproduce the reconstruction of the environment perceived by human echolocation and represented in the human mind.

Appendix A

Sound absorption

Sound is produced by the vibration of objects. The object or substance through which the sound propagates is defined means and it can be solid, liquid, or gas.

Sound waves move from the origin point (source) to the auditor's direction. Particularly, when the object vibrates it transmits that vibration to the adjacent particles of the means, no single particle travels from the source to the ear. Each means particle, located close to the source, is moved from its equilibria position, and forced to move; and it will transmit the energy to the near particles enabling sound propagation. The particles of the means do not move forward on their own: the sound propagation can be described with an undulatory motion given by the acoustic wave.

The air is surely the most common propagation means; when an object vibrates, it pushes and pulls the adjacent air. The compressed air starts to move from the source, and then returns to the equilibrium status, a low-pressure area is created, defined as "refraction". This oscillating movement repeats itself many times, generating diverse compression and refraction areas in the air. The alternating of these areas allows the acoustic wave to propagate through the means. The high- and low-pressure zones in the means are related to the density of the particles in a certain volume. The higher the particles number the higher the pressure. It can be said that sound propagation is allowed by the pressure variability in the means.

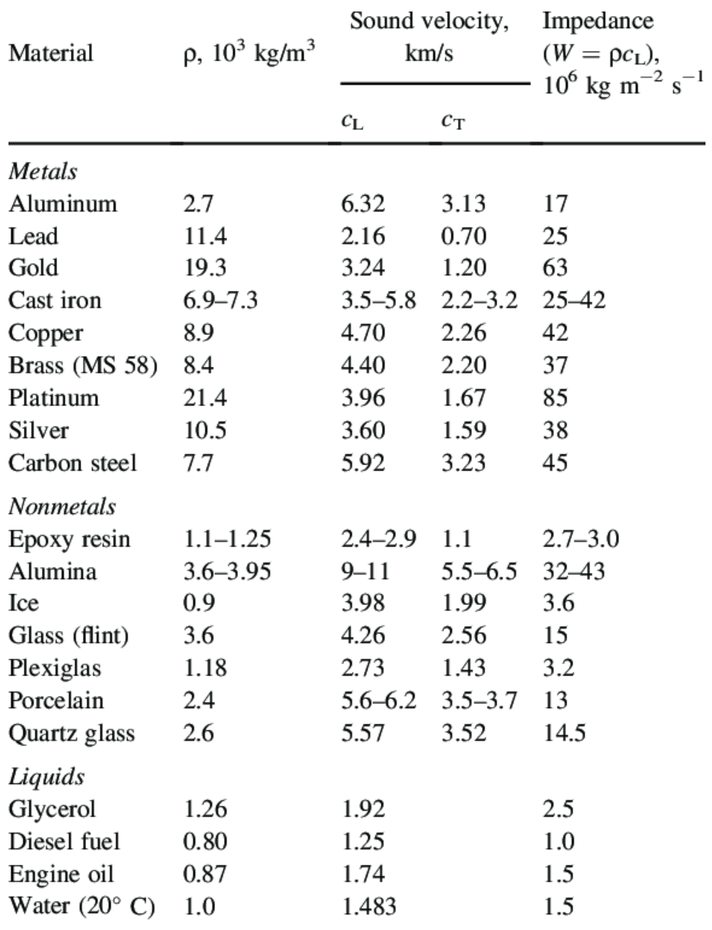
The presence of a means is necessary for sound propagation. Since it is a mechanical wave, the fundamental condition for sound propagation is the presence of a physical means (water, air, metal, etc.). Sound cannot travel in the void. The intensity is generally measured in decibels (dB).

An important phenomenon of the sound propagation is the reflection that is determined when the sound wave meets an obstacle and return backward. There are two waves: the incident wave and the reflected wave. The reflection can be characterized as echo or as reverberation.

In physics and acoustics, the echo is a phenomenon produced by the sound waves reflection caused by an obstacle and perceived by the hearer affected by a certain delay. A human listener has the ability to distinguish the sound wave reflection from the original sound, in case of echo. The echo can be described as a wave that is reflected by a propagation means discontinuity and returns with an intensity and delay sufficient to be perceived. Th echo can be noticed in useful implementations, for instance in the sonar system.

The reverberation can be explained at the listener's ear as a superposition of the incident wave and the reflected wave. In the case of echo, they can be distinguished. To have an echo, it is necessary that the distance between the source and the listener is about at least 17 metres at the conditions of ambient temperature at 20°C and 340 m/s as speed of sound. To distinguish the two waves, incident and reflected, it is needed to have a time delay of 1/10 of second in between. This interval in the air corresponds to the distance of 34 metres, that can be translated in 17 metres from the sound source and 17 metres to achieve the hearer. If the distance is minor, the phenomenon is the reverberation. In other words, in case of echo the delay is not be lower than 1/10 of a second; under this threshold the phenomenon is called reverberation.

Moreover, the sound propagates in a specific means at a constant speed. As during a thunderstorm, the thunder is heard after having seen the lighting: the speed of sound is lower than the speed of light, and its velocity lays on the physical properties of the means. Mainly, the speed of sound is direct function to the temperature of the transmitting medium. The speed of sound in gasses can be expressed by the following formula: $v_{sound} = \sqrt{\frac{\gamma RT}{M}}$ where γ is the adiabatic constant, R gas constant, M molecular mass of gas, T absolute temperature. The higher the temperature, the faster the sound. For instance, the velocity in the air at 0°C is 331m/s, while at 22°C is 344m/s. Additionally, the sound velocity decreases from a solid means to a gas; hence it strongly depends on the density of the material.



Material	ρ, 10^3 kg/m^3	Sound velocity, km/s		Impedance ($W = \rho c_L$), 10^6 kg m^{-2} s^{-1}
		c_L	c_T	
Metals				
Aluminum	2.7	6.32	3.13	17
Lead	11.4	2.16	0.70	25
Gold	19.3	3.24	1.20	63
Cast iron	6.9–7.3	3.5–5.8	2.2–3.2	25–42
Copper	8.9	4.70	2.26	42
Brass (MS 58)	8.4	4.40	2.20	37
Platinum	21.4	3.96	1.67	85
Silver	10.5	3.60	1.59	38
Carbon steel	7.7	5.92	3.23	45
Nonmetals				
Epoxy resin	1.1–1.25	2.4–2.9	1.1	2.7–3.0
Alumina	3.6–3.95	9–11	5.5–6.5	32–43
Ice	0.9	3.98	1.99	3.6
Glass (flint)	3.6	4.26	2.56	15
Plexiglas	1.18	2.73	1.43	3.2
Porcelain	2.4	5.6–6.2	3.5–3.7	13
Quartz glass	2.6	5.57	3.52	14.5
Liquids				
Glycerol	1.26	1.92		2.5
Diesel fuel	0.80	1.25		1.0
Engine oil	0.87	1.74		1.5
Water (20° C)	1.0	1.483		1.5

Table 12. Sound velocity strongly relies on the density of the means. The figure, selected from [33], reports some of the most common materials in industry.

The substances can be additionally categorized in relation to how they interact with waves. Some materials have the property to widely absorb sound waves and lessen their energy, while others reflect them for a high percentage.

The sound-absorption is the ability of a material to keep a percentage of the sound that achieves it. The phenomenon of the acoustic absorption can be explained by observing the nature of sound-absorbing materials, that are characterized by a sponge and porous structure. Consequently, they have a relevant percentage of holes, in which the air particles are moved by the propagation of the sound waves that achieve the surfaces of these materials. The sound waves are mainly reflected and mirrored into the holes, the air particles motion is transformed into heat and the energy is scattered.

The characteristic of the sound-absorbing panels to allow the air passage is connected to the concept of the Resistivity of Flow, that measures the ability of the mater to able the transit of the air, even opposing resistance to that. If the material permitted the entire flow of the air without friction, the sound wave dissipation in heat form would not happen. A good sound-absorbing materials is able to balance both the needs: enhancing the air flow in a series of holes and channels and opposing resistance to the passage. From this point, it is clear that a rigid, thin, dense material cannot offer a good sound-absorption.

However, the most internal layers of the walls in standard house room are not composed by those material. The sound isolation, if present, is provided in the middle layers; hence the internal layers are usually hard and smooth walls, with a consistently low capacity of sound absorption. In the next paragraph, a deeper explanation of the sound absorption characterization of different materials is presented.

The performance value of a sound-absorbing material is indicated as an alfa coefficient, and it is comprehended between 0 and 1. An index higher than 1 is not physically feasible, because that would mean the possibility to keep more sound than the received.

The value can be indicated in the datasheet in different manners. One example is the curve of sound-absorption. The following example shows how it is possible to read the single coefficient for each frequency

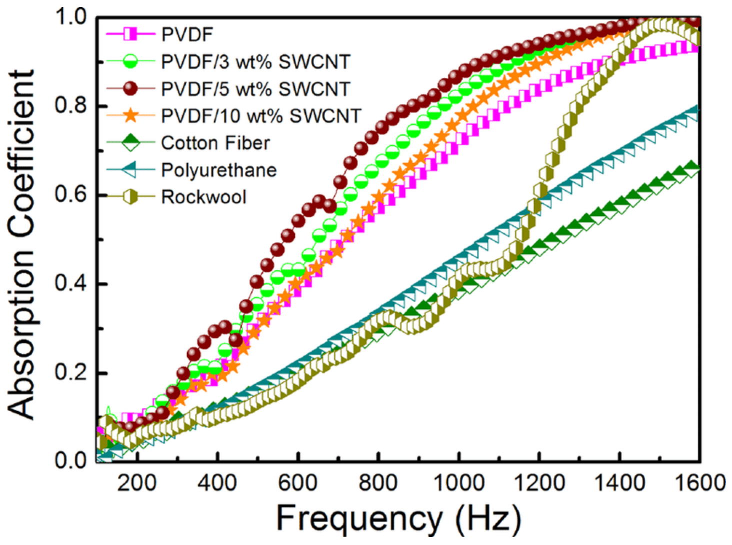


Figure 51. Curves of sound-absorption with a coefficient value (α) for each frequency [34]

Moreover, a diverse index considers the values obtained by the Sabine's Method: each value is the "equivalent area of acoustic absorption" or "total absorption" for each frequency.

The UNI EN ISO 11654:1998 introduced the sound-absorption classes to classify the sound-absorbing material with performances indexes between A and E. It translates the sound absorption coefficients depending on the frequency to a single evaluation index.

Sound Absorption Classes	Sound Absorption Coefficient Range
A	0.90–1.00
B	0.80–0.85
C	0.60–0.75
D	0.30–0.55
E	0.15–0.25
F	0.00–0.10

Table 13. Sound-absorption classes [35]

Sound-absorbent materials

The acoustic foam panels are a perfect example of sound absorbent material and is shown in the following picture.

Figure 52. Acoustic foam panel

The foam panels, generally, can absorb the 80% of the frequency between 125Hz and 2kHz, a considerable stripe of the audible: it depends on the softness and porosity of the acoustic foam, but also on the typology of shape and position.

Some of the most reflective surfaces on the market are named in the following rows: the marble, for instance. Because of its hardness of 3 on the Mohs scale, it can be named in this list; however, the marble could be porous, hence it can absorb a percentage of the sound waves. The granite is one of the most reflective materials because of its density and resistance; its hardness is 6.5 on the Mohs scale and its density is 76 kg/ft^3: these properties make it denser and harder than the marble. The granite is porous, as the marble, however when it is employed in the houses, a seal is foreseen, therefore, its surface is perfectly reflective.

The clay bricks replicate the stone behaviour in terms of interaction with the sound waves. Although, they are considerably less dense than the marble, the sound waves are reflected successfully. However, a rough clay bricks wall absorbs between the 3% and 5% of all the frequencies. The ceramic tiles have an incredibly low absorption percentage, even if as hard as the above materials. They can be mentioned between the reflective substances, because of an absorption of the 2% of all the frequencies.

The smooth concrete is compact, hard and it polymerizes: these properties make it a perfect material for buildings. Its answer to the sound waves depends on the different surficial treatments: the rough concretes sound absorption is consistently higher than the one of the painted concretes. The variant of the clinker concrete is particularly harsh and can consume between the 10% and 60% of all the frequencies.

Besides, the concrete walls are usually revested by a layer of plaster. Its texture lets an absorption between 5% and 10% of the frequencies. Different behaviour for the metallic materials: their clear shiny surface is highly reflective. The sound absorption coefficient is around the 0,03%, that means the reflection is about the 97% of the sound.

The windows, constituted by glass, or mirrors have a similar behaviour to metallic plate surfaces. The sound absorption coefficient is around 0.03, it relies on glass characteristics: for instance, a glass plate of four millimetres of thickness can absorb till the 30% of the low frequency sound waves and the 2% of high frequency sound waves. Generally, the glass can reflect between the 90% and 98% of the frequencies. The plastic has a similar behaviour considering that the surfaces are compact, smooth, and solid: it can reflect between the 95% and 100% of the sound.

The wood can absorb a good percentage: the plywood can reflect between the 70% and 91% of the sound; instead, the solid wood increases these percentages to 86% and 92%. It depends on hardness, density, and surface. Moreover, the water can reflect the sound in specific conditions of calm water and plater water surface. Also, the tents, if heavy and compact, can reflect the sound.

From the evaluation of different materials and scenarios behaviour about the sound absorption, it is possible to interpreter and predict the possible sound reflections. These considerations let to get data consciously and effectively.

References

[1] Smith L., Gasser M. The development of embodied cognition: Six lessons from babies. Artificial life 2005.

[2] Senocak A., Ryu H., Kim J., Kweon I. S. Less Can Be More: Sound Source Localization with Classification Model. WACV 2022.

[3] Senocak A., Oh T.H., Kim J., Yang M. H., Kweon I. S., Learning to Localize Sound Surce in Visual Scene. CVPR 2018, TPAMI 2020.

[4] Owens A., Efros A. A. Audio-Visual Scene Analysis with Self-Supervised Multisensory Features. ECCV 2018.

[5] Arandjelovic R., Zisserman A. Look, Listen and Learn. ICCV 2017.

[6] Arandjelovic R., Zisserman A. Objects that sound. ECCV 2018.

[7] Tian Y., Shi J., Li B., Duan Z., Xu C. Audio-Visual Event Localization in the Wild. CVPR Workshop 2019.

[8] Aytar Y., Vondrick C., Torralba A. SoundNet: Learning Sound Representations from Unlabeled Videobat. NIPS 2016.

[9] Gao R., Chen C., Al-Halah Z., Schissler C., Grauman K. VisualEchoes: Spatial Image Representation Learning through Localization. ECCV 2020.

[10] Jesper Haahr Christensen, Sascha Hornauer, and Stella Yu. Batvision: Learning to see 3d spatial layout with two ears. ICRA, 2020.

[11] Kranti Kumar Parida, Siddharth Srivastava, and Gaurav Sharma: Beyond Image to Depth: Improving Depth Prediction using Echoes. CVPR, 2021.

[12] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, Y. Zhang. Matterport3D: Learning from RGB-D Data in Indoor Environments. International Conference on 3D Vision (3DV) 2017.

[13] Straub J., Whelan T., Ma L., Chen Y., Wijmans E., Green S., Engel J. J., Mur-Artal R., Ren C., Verma S., Clarkson A., Yan M., Budge B., Yan Y., Pan X., Yon J., Zou Y., Leon K., Carter N., Briales J., Gillingham T., Mueggler E., Pesqueira L., Savva M., Batra D., Strasdat H. M., De Nardi R., Goesele M., Lovegrove S., Newcombe R. The Replica Dataset: A Digital Replica of Indoor Spaces. 2019

[14] Savva, M., Kadian, A., Maksymets, O., Zhao, Y., Wijmans, E., Jain, B., Straub, J., Liu, J., Koltun, V., Malik, J., Parikh, D., Batra, D. Habitat: A Platform for Embodied AI Research. ICCV 2019.

[15] A. Zunino, M. Crocco, S. Martelli, A. Trucco, A. Del Bue, and V. Murino, Seeing the sound: A new multimodal imaging device for computer vision. IEEE International Conference on Computer Vision Workshop (ICCVW) 2015.

[16] E. Tracy, N. Kottege. CatChatter: Acoustic Perception for Mobile Robots. IEEE Robotics and Automation Letters, Vol. 6, No. 4, 2021.

[17] I. Eliakim, Z. Cohen, G. Ksa, and Y. Yovel. A fully autonomous terrestrial bat-like acoustic robot. PLOS Computational Biology, vol. 14, 2018.

[18] Steckel J., Peremans H. BatSLAM: Simultaneous Localization and Mapping Using Biomimetic Sonar. PLOS 2013.

[19] Supa M, et al. 'facial vision'. The perception of obstacles by the blind. Am. J. Psychol 1944.

[20] Kellogg WN. Sonar system of the blind. Science 1962.

[21] Rice C. Human echo perception. Science 1967.

[22] Liu P.Y., Lam E. Y., Wu Y. C. Image Reconstruction Using Deep Learning. https://arxiv.org/ftp/arxiv/papers/1809/1809.10410.pdf

[23] Huang G., Liu Z., Weinberger K., van der Maaten L., Densely Connected Convolutional Networks. CVPR 2017.

[24] Zhang Y., A Better Autoencoder for Image: Convolutional Autoencoder. http://users.cecs.anu.edu.au/~Tom.Gedeon/conf/ABCs2018/paper/ABCs2018_paper_58.pdf

[25] Alhashim I., Wonka P., High Quality Monocular Depth Enstimation via Trasfer Learning. 2018 https://arxiv.org/abs/1812.11941

[26] Godard C., Mac Aodha O., Brostow G. J. Unsupervised Monocular Depth Enstimation with Left-Right Consistency. CVPR 2017.

[27] Laina I., Rupprecht C., Belagiannis V., Tombari F., Navab N. Deeper Depth Prediction with Fully Convolutional Residual Networks. 3DV 2016.

[28] Godard C., Mac Aodha O., Firman M., Brostow G. Digging Into Self-Supervised Monocular Depth Estimation. ICCV 2019.

[29] Ranftl R., Bochkovskiy A., Koltum V. Vision Transformers for Dense Prediction. 2021 https://arxiv.org/abs/2103.13413

[30] Fu H., Gong M., Wang C., Batmanghelich K., Tao D. Deep Ordinary Regression Network for Monocular Depth Estimation. CVPR 2018.

[31] Milford M. J., Wyeth G. F., Prasser D. RatSLAM: A Hippocampal Model for Simultaneous Localization and Mapping. ICRA 2004.

[32] He K., Zhang X., Ren S., Sun J. Deep Residual Learning for Image Recognition, 2015 https://arxiv.org/abs/1512.03385

[33] Ono K., Erhard A. Ullmann's Encyclopedia of Industrial Chemistry. 2011

[34] Rahimabady M., Statharas E. C., Yao K., Mirshekarloo M. S., Chen S., Hock Tay F. E. National University of Singapore Hybrid local piezoelectric and conductive functions for high performance airborne sound absorption. 2017

[35] Hassan T., Jamshaid H., Mishra R., Khan M. Q., Petru M., Novak J., Choteborsky R., Hromasová M. Czech University of Life Sciences Acoustic, Mechanical and Thermal Properties of Green Composites Reinforced with Natural Fibers Waste. 2020

[36] L. Li. Time-of-Flight Camera—An Introduction. Texas Instruments-Technical White Paper, 2014.

[37] https://github.com/PortAudio/portaudio

[38] Simonyan K., Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition. CS 2015.

[39] Teng S., et al. Ultrafine spatial acuity of blind expert human echolocators. Exp Brain Res 2012.

[40] Thaler L., De Vos H. P. J. C., Kish D., Antoniou M., Baker C. J., Hornikx M. C. J. Human Click-Based Echolocation of Distance: Superfine Acuity and Dynamic Clicking Behaviour. JARO 2019.

[41] Norman L. J., Dodsworth C., Foresteire D., Thale L. Human click-based echolocation: Effects of blindness and age, and real-life implications in a 10-week training program. PLOS 2021.

[42] Sohl-Dickstein J., Teng S., Gaub B. M., Rodgers C. C., DeWeese M. R., Harper N. S. A device for human ultrasonic echolocation. IEEE Trans Biomed Eng 2015.

[43] Cheng Z., Sun H., Takeuchi M., Katto J. Deep Convolutional AutoEncoder-based Lossy Image Compression. CV 2018

[44] Mao X., Shen C., Yang Y. Image Restoration Using Convolutional Auto-encoders with Symmetric Skip Connections. CV 2016

[45] Alqahtani A., Xie X., Deng J., Jones M.W. A DEEP CONVOLUTIONAL AUTO-ENCODER WITH EMBEDDED CLUSTERING. IEEE 2018.

[46] Chen M., Shi X., Yin Zhang Y., Wu D., Guizani M. Deep Feature Learning for Medical Image Analysis with Convolutional Autoencoder Neural Network. IEEE 2021.

[47] David E., Netanyahu N. S. DeepPainter: Painter Classification Using Deep Convolutional Autoencoders. CV 2017.

[48] Eitan O., Weinberg M., Danilovich S., Barkai Y., Assa R., Yovel Y. Functional daylight echolocation in highly visual bats. Current Biology 2022.

[49] Salles A. Bats: Vision or echolocation, why not both? Current Biology 2022.

[50] Thaler L., Reich G. M., Zhang X., Wang D., Smith G. E., Tao Z., Abdullah R., Cherniakov M., Baker C. J., Kish D., Antoniou M. Mouth-clicks used by blind expert human echolocators – signal description and model based signal synthesis. PLOS Computational Biology 2017.

[51] Brancati R., Cosenza C., Niola V., Savino S. Experimental Measurement of Underactuated Robotic Finger Configurations via RGB-D Sensor. 27th International Conference on Robotics in Alpe-Adria-Danube Region 6-8 2018.

[52] Sansoni G., Trebeschi M., Docchio F. State-of-The-Art and Applications of 3D Imaging Sensors in Industry, Cultural Heritage, Medicine, and Criminal Investigation. PubMed 2009.

[53] https://miro.medium.com/max/1400/1*hkYlTODpjJgo32DoCOWN5w.webp

Thank you for the attention

I would like to express my infinite gratitude to my supervisor, Professor David Scaradozzi, who has supported me in all my needs, from a technical side, but mostly important, also at a personal level. He has been encouraging me since my first experience abroad and allowed me to live two of the most life-changing experiences I have ever gone through. He has always solved my questions and requests with enthusiasm and supportive spirit, and he has always provided me with anything I needed along the path. I thank him for believing in me.

A special thanks to Doctor Nicholas Lawrance, who believed in me and my capacities and guided me throughout the first part of this project. He has always been present and available at any uncertainty and I am delighted to have been tutored by a researcher with incredible intelligence and preparation like him, and positive spirit in encountered any request.

A particular thanks to Doctor Lionel Ott, who led me through any step during the second part of my thesis. I am more than grateful to have been led by a personality with huge passion and intellect. I am pleased he dedicated his diligence to any aspect and to solve any doubt I stepped into. He left me enchanted by her always spot-on predictions and comments.

I would like to convey my appreciations to Professor Roland Siegwart, who accepted me in his incredible lab and allowed me to meet incredibly smart and hard-working researchers and students.

I also have to express my gratitude to my boyfriend Giacomo, always the major hearty and warm supporter and wearing the shoes of a guardian angel of my personal fulfilment and happiness.

I would also like to thank my family who always back me and defend my dreams with all their possibilities.