Politecnico di Torino

Politecnico di Torino

Master's Degree in Biomedical Engineering
A.Y. 2024/2025
December 2025 Graduation Session

Automated Deep Learning of Primary Progressive Aphasia (PPA) variants from cognitive-test voice recordings and T1-weighted MRI data

Supervisors:
Prof. Molinari Filippo

Candidate:
Alice Corini

Co-Supervisors:
Prof. Salvi Massimo
Prof. Filippi Massimo
Prof.ssa Agosta Federica
Dott.ssa Basaia Silvia

Summary

Table of Figures

Table of Tables

Acronyms

1D-CNN = One-Dimensional Convolutional Neural Network
2D-CNN = Two-Dimensional Convolutional Neural Network
3D-CNN = Three-Dimensional Convolutional Neural Network
AC = Alternating Current
AD = Alzheimer's Disease
Adam = Adaptive Moment Estimation
AI = Artificial Intelligence
ALS = Amyotrophic Lateral Sclerosis
ANNs = Artificial Neural Networks
ASR = Automatic Speech Recognition
BA = Balanced Accuracy
BCE = Binary Cross-Entropy
BETO = Bidirectional Encoder Representations from Transformers for Spanish
BiLSTM = Bidirectional Long Short-Term Memory
BiRNN = Bidirectional Recurrent Neural Network
BERT = Bidirectional Encoder Representations from Transformer
BOLD = Blood Oxygen Level Dependent
BPTT = Back Propagation Through Time
bvFTD = Behavioral Variant Frontotemporal Dementia
CCE = Categorical Cross-Entropy
CM = Confusion Matrix
c-MCI = converting Mild Cognitive Impairment
CNN = Convolutional Neural Network
CPMG = Carr Purcell Meiboom Gill
CREMA = Crowd-sourced Emotional Multimodal Actors
CSE = Conventional Spin Echo
CSF = Cerebrospinal Fluid
CV = Cross Validation
CT = Computed Tomography
DT = Decision Tree
DTI = Diffusion Tensor Imaging
EA = Evolutionary Algorithms
EMO-DB = Berlin Database of Emotional Speech
FFT = Fast Fourier Transform
FID = Free Induction Decay
fMRI = Functional Magnetic Resonance Imaging
FN = False Negative
FoV = Field of View

FP = False Positive
FTD = Frontotemporal Dementia
GELU = Gaussian Error Linear Unit
GSM = Groupe Spécial Mobile
HC = Health Control
k-NN = K-Nearest Neighbors
LOO-CV = Leave-One-Out Cross-Validation
LOSO-CV = Leave-One-Subject-Out Cross-Validation
Lr = Learning Rate
LSO-CV = Leave-Subject-Out Cross-Validation
LSTM = Long Short-Term Memory
lvPPA = Logopenic Variant Primary Progressive Aphasia
MAE = Mean Absolute Error
MCI = Mild Cognitive Impairment
MDD = Major Depressive Disorder
MFCC = Mel-Frequency Cepstral Coefficients
MLP = Multilayer Perceptron
MMSE-DS = Mini-Mental State Examination for Dementia Screening
MNI = Montreal Neurological Institute
MSE = Mean Squared Error
MRI = Magnetic Resonance Imaging
nfvPPA = Nonfluent Variant Primary Progressive Aphasia
NMR = Nuclear Magnetic Resonance
NPV = Negative Predicted Value
OOV = Out-of-Vocabulary
PD = Parkinson's Disease
PET = Positron Emission Tomography
PPA = Primary Progressive Aphasia
PPV = Positive Predicted Value
RAVDESS = Ryerson Audio-Visual Database of Emotional Speech and Song
RBF = Radial Basis Function
ReLU = Rectified Linear Unit
RF = Radio Frequencies
RMSProp = Root Mean Square Propagation
RMSE = Root Mean Squared Error
RNNs = Recurrent Neural Networks
SAVEE = Surrey Audio Visual Expressed Emotion
SE = Spin Echo
s-MCI = stable Mild Cognitive Impairment
SGD = Stochastic Gradient Descent
SPEC = log-mel spectrograms
STFT = Short-Time Fourier Transform

SVM = Support Vector Machine
svPPA = Semantic Variant Primary Progressive Aphasia
Tanh = Hyperbolic Tangent Function
TESS = Toronto Emotional Speech Database
TE = Echo Time
TN = True Negative
TP = True Positive
TR = Repetition Time
VBM = Voxel-Based Morphometry
W2V = Word to Vector
WAB = Western Aphasia Battery
Wav2vec = Waveform to Vector
WBCE = Weighted Binary Cross-Entropy

Abstract

Primary Progressive Aphasia (PPA) is a neurodegenerative disorder characterized by a gradual decline in language abilities, caused by localized atrophy in specific cortical regions. It manifests in three variants: non-fluent (nfvPPA), semantic (svPPA), and logopenic (lvPPA), each associated with distinct linguistic profiles and atrophy patterns. Standardized language batteries, together with structural Magnetic Resonance Imaging (MRI) quantification of atrophy, are currently the reference methods for clinical diagnosis. Recently, deep learning approaches have shown promise in supporting diagnosis by automatically extracting patterns from input data.

The aim of this study was to develop deep learning models for automatic classification of PPA variants by integrating two clinical modalities: voice recordings collected during clinician-administered cognitive tests and volumetric 3D T1-weighted structural MRI.

Data were collected at the IRCCS San Raffaele Scientific Institute and included 94 PPA patients (38 nfvPPA, 36 svPPA, 20 lvPPA) and 91 healthy controls (HC). Of these, 81 PPA and 38 HC completed the Picnic picture description task from the Western Aphasia Battery (WAB), designed to evaluate connected speech production. Moreover, 90 PPA (38 nfvPPA, 33 svPPA, 19 lvPPA) and 82 HC underwent structural MRI.

Audio signals underwent preprocessing for noise removal, normalization, and segmentation. The resulting fragments were converted into log-Mel spectrograms and used to train a two-dimensional Convolutional Neural Network (2D-CNN).
In a second approach, a 2D-CNN combined with a Bidirectional Long Short-Term Memory (BiLSTM) network was implemented to capture both spectral and temporal dependencies of speech.
Finally, a Wav2Vec 2.0 model pre-trained on Italian speech, with the last two layers unfrozen, was fine-tuned on the PPA dataset using one-dimensional audio and additional classification layers.

MRI images were preprocessed and cropped into the regions most affected by atrophy across PPA variants (left hemisphere) and then used to train a 3D-CNN.
A late fusion strategy was subsequently applied to combine the predictions of the audio and MRI networks.

All architectures were evaluated for both multiclass and binary classification tasks. At the multiclass level, the best-performing models were the 2D-CNN with BiLSTM and Wav2Vec, achieving balanced accuracies of 0.62 and 0.75, respectively, suggesting

that more complex architectures are beneficial for more challenging tasks. At the binary level, the simpler 2D-CNN achieved the best performance, indicating that less complex models may be sufficient for simpler classification tasks. Specifically, it performed well in distinguishing HC from PPA overall, as well as in comparisons with single PPA variants. Binary comparisons between PPA variants were more difficult, reflecting the increased challenge of discriminating between specific subtypes.

The 3D-CNN model achieved a balanced accuracy of 0.69 at the multiclass level and comparable results in binary comparisons, performing better in distinguishing between variants due to its ability to identify localized atrophy patterns.

A late fusion strategy, applied only at the multiclass level, further improved performance: the fusion of 2D-CNN with BiLSTM and 3D-CNN achieved a balanced accuracy of 0.83, while combining Wav2Vec and 3D-CNN reached 0.94.

These results demonstrate that deep learning can effectively support automatic PPA diagnosis, with performance further enhanced by integrating audio and imaging data.

1. Primary Progressive Aphasia

Primary progressive aphasia (PPA) is a clinical neurodegenerative syndrome involving progressive deterioration of language ability, linked to cerebral atrophy.[1] Unlike classic forms of aphasia, which develop suddenly following strokes, ischemic attacks or brain injuries, PPA manifests itself gradually, precisely because it is caused by a neurodegenerative process.

PPA deficits remain confined to the language area in the initial phase of the disease; subsequently, additional cognitive symptoms may arise in the more advanced stages of the disease, despite linguistic dysfunction being the most salient feature.[2]

PPA is considered a subtype of fronto-temporal dementia (FTD), a group of diseases characterized by neuronal degeneration affecting the temporal and frontal lobes. FTD targets brain areas responsible for personality, behavior, language learning, motivation, abstract thinking, and executive functions. Since 2011, FTD has been divided into two variants, the behavioral variant (bvFTD) and PPA.[3]

In 2011, three variants of PPA were introduced: the non-fluent/agrammatic variant (nfvPPA), the semantic variant (svPPA), and the logopenic variant (lvPPA), diagnosed based on specific characteristics of language and speech, characteristics of each subtype.[1]

1.1 Historical Background

The concept of PPA originated over a century ago, in 1892, when Pick, a Czech neurologist, described for the first time the clinical profile of a patient with a progressive language disorder, associated with atrophy of the frontal and temporal lobe of the left hemisphere.

Pick focused only on detailed clinical observations, as histological techniques were limited. Subsequently, Alois Alzheimer, in 1911, analysed histologically Pick's clinical cases and found the cellular changes characterizing the disease, including the presence of argyrophilic cytoplasmic inclusions in neurons.

In 1923, Gans introduced the concept of "Pick's atrophy" to describe patients with frontotemporal atrophy.

Later, the studies were continued by Pick's students, Onari and Spatz, who further investigated Pick's pathology, also building on Alzheimer's studies, defining "Pick's bodies", argyrophilic cytoplasmic inclusions, and "Pick's cells", enlarged and distorted neurons. Thus, a disease was defined both through the description of clinical signs and by evaluating neuropathological alterations.[4] [5]

Subsequently, research in this field underwent a period of stagnation until the 1980s, when Mesulam, in 1982, described the aphasia of six patients aged between 17 and 69

years. One of them underwent an autopsy, showing that the brain had a nonspecific loss of neurons, but did not show Alzheimer's disease or vascular alterations, suggesting the existence of a disease independent of Alzheimer's, which Mesulam defined as "slowly progressive aphasia".[5]

Only in 1992 did Mesulam and Weintraub coin the term PPA, to provide a better representation of the disease, including from a terminological perspective. Indeed, the term aphasia indicated a language disorder, progressive indicated that the disorder worsened over time, and primary indicated that aphasia was the main and initial symptom of the disease.

For about two decades, cases of PPA were classified as semantic dementia or nonfluent progressive aphasia. Semantic dementia was first described in 1975 by Warrington and later by Snowden et al., who introduced the term. In the early 1990s, Hodges and colleagues provided a complete clinical characterization of semantic dementia. The nonfluent variant, instead, was first described by Grossman et al. in 1996.[5]

The third variant, the logopenic variant, was introduced in 2004 by Gorno-Tempini et al., as it encompassed linguistic features that did not fit into the previous binary subdivision of PPA. [6]

A major milestone in the history of PPA occurred in 2011, when a group of clinical experts met with the aim of defining the diagnostic criteria for PPA and its three main clinical variants criteria that are still in use today.[1]

The timeline of the history of PPA is illustrated in Figure 1.

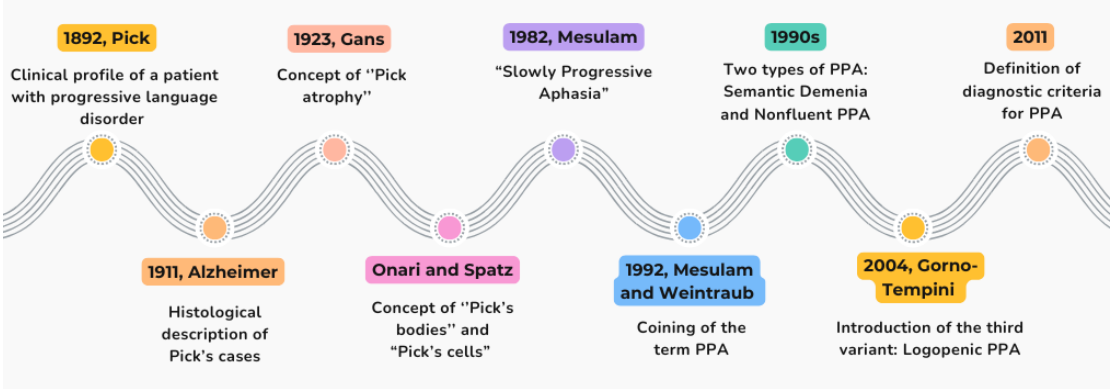

Figure 1: Timeline: From Pick's first observations to the modern diagnostic criteria for PPA.

1.2 Epidemiology and Ethology

Cases of PPA have been reported worldwide, with no evidence of specific geographic clustering. The overall prevalence of PPA, including all variants, is estimated to range between 3 and 7 cases per 100,000 people, depending on the population studied. The annual incidence is estimated at approximately 1.14 cases per 100,000 person-years, compared with 35.7 cases per 100,000 person-years for Alzheimer's disease.[7,8]

Regional variations may be influenced by disease awareness and access to appropriate diagnostic tools.[8]

In memory clinics, PPA accounts for approximately 0.5–2.5% of all patients diagnosed with a neurodegenerative disorder. The mean age at onset is around 60 years, with a range of 45–70 years, and the disease duration varies from 4 to 14 years, with an average of approximately 8 years.[8]

Regarding gender distribution, the literature shows heterogeneous results. Some studies, such as Turcano et al. (2024), report a slightly higher prevalence in women (60% vs. 40%)[9], while others, like Ulugut et al. (2022), highlight variant-specific differences, with a male predominance in the semantic variant and a female predominance in the non-fluent/agrammatic variant.[10]

Overall, no clear demographic, environmental, or socioeconomic risk factors have been identified for PPA.[8]

Among recent studies, Turcano et al. (2024) analyzed the incidence of PPA in Olmsted County, Minnesota, from January 1, 2011, to December 31, 2022. The study included all adults over 18 years of age residing in the county and calculated PPA incidence as the number of new cases per 100,000 inhabitants. It is one of the first epidemiological studies to separately evaluate the three main PPA variants, reporting the following incidence rates:

- Overall PPA incidence: 0.56 cases per 100,000 person-years (95% CI: 0.24–1.10)
- nfvPPA: 0.14 cases per 100,000 person-years (95% CI: 0.02–0.55)
- lvPPA: 0.21 cases per 100,000 person-years (95% CI: 0.04–0.61)
- svPPA: 0.21 cases per 100,000 person-years (95% CI: 0.04–0.61)[8,9]

1.3 Brain regions involved in language

The brain is one of the most complex organs in the human body. It is composed of approximately 100 billion neurons that communicate with each other through numerous connections and synapses. It is made up of nervous tissue and controls various task-evoked responses, movement, senses, emotions, language, communication, thinking, and memory.[11]

Among the numerous functions carried out by the brain, one of the most relevant for PPA is language.[12] The brain areas associated with language are the following:

- **Frontal lobe:** the largest lobe, located at the front of the cerebral hemispheres, which controls speech and language. Within the inferior frontal gyrus lies Broca's area, essential for producing articulated speech and grammar. Lesions in this region cause non-fluent or agrammatic speech.[11]
- **Parietal lobe:** located posteriorly to the frontal lobe and above the temporal lobe. It is divided into two regions, but only the posterior parietal lobe is involved in

language, specifically the inferior parietal lobe. This region includes the Geschwind territory (angular gyrus and supramarginal gyrus), which is involved in lexical processing, reading, and writing. The Geschwind territory plays a crucial role as it connects the two main areas involved in language: Broca's area and Wernicke's area. Damage to these regions may impair reading comprehension and semantic processing. [12]

- **Temporal lobe:** located behind the frontal lobe and beneath the parietal lobe. The posterior superior temporal lobe contains Wernicke's area, essential for the comprehension of spoken and written language. The anterior temporal regions are responsible for accessing word meaning. Lesions in this area may lead to deficits in word retrieval and semantic knowledge; typically, the patient speaks fluently, but their words and sentences lack meaning. [12]

Figure 2 illustrates the anatomical positions of these lobes.

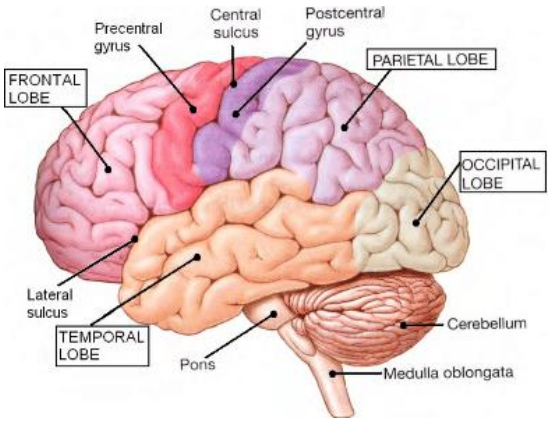

Figure 2: Illustration of the cerebral lobes.[13]

The three areas essential for language: Broca's area, Wernicke's area, and the Geschwind territory, are interconnected by the arcuate fasciculus and additional parietal pathways, which enable the exchange of information between language comprehension and production. These three areas and their connections are illustrated in Figure 3.

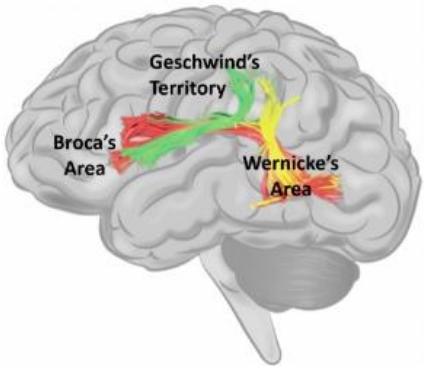

Figure 3: Illustration of Broca's area, Wernicke's area, and Geschwind territory.[14]

1.4 Neuropathological Brain Changes According to PPA Variant

Neuropathology of PPA is primarily characterized by progressive cortical atrophy, primarily affecting the left hemisphere. Depending on the brain regions involved, three main variants of PPA can be distinguished. [15,16]

- svPPA: characterized by cortical thinning in the left anterior temporal lobe, which may extend to the left posterior temporal lobe as the disease progresses.[17]
- lvPPA: shows milder atrophy, mainly localized in the left middle posterior temporal gyrus and the left anterior temporal cortex.[17]
- nfvPPA: exhibits predominant atrophy in frontal regions, particularly in Broca's area. Other affected regions include more dorsal and medial areas of the posterior frontal cortex, such as the left premotor cortex, supplementary motor area, and primary motor cortex. [16,17]

The following Figure 4 shows the regions of greatest cortical thinning for each PPA variant. The colormap represents statistical significance: blue indicates areas that are significant but less rigorous, whereas yellow highlights regions with highly significant cortical thinning. From the figure, it can be observed that:

- lvPPA: displays two main clusters located at the left temporo-parietal junction and the left anterior temporal cortex.[17]
- svPPA: shows clusters affecting most of the temporal lobes, with lesser involvement of the inferior frontal gyrus.[17]
- nfvPPA: exhibits extensive clusters in frontal regions, with partial extension into temporal and parietal areas.[17]

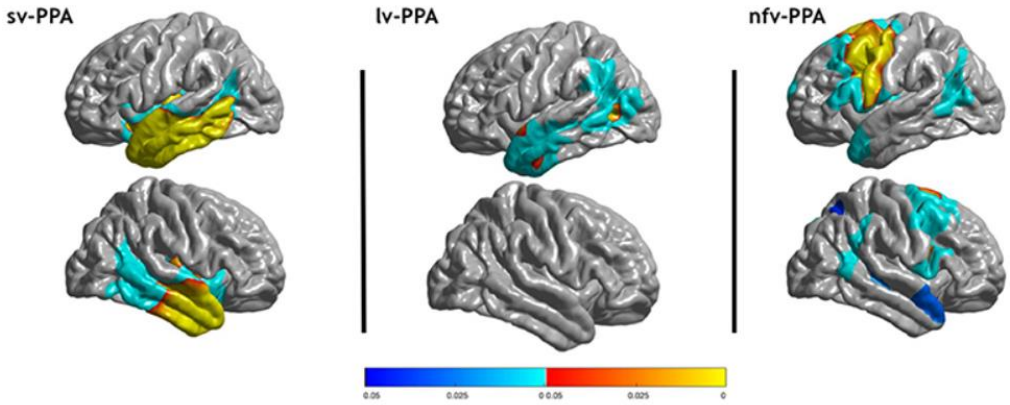

Figure 4: Areas of significantly reduced cortical thickness in sv-PPA, lv-PPA and nfv-PPA, compared to healthy controls. [17]

1.5 Linguistic Characteristics of PPA Speech

Based on the distribution of brain neurodegeneration, the main variants of PPA differ in their linguistic expression:

- **Agrammatic/non-fluent variant**
  Patients manifest an agrammatical language, with omission of pronouns, articles, and prepositions. The sentences pronounced are short and often have an incorrect syntactic order, resulting in a non-fluent language. Grammatical comprehension is compromised, especially for sentences that are long and complex. Grammatical difficulty is manifested even more in writing. Comprehension and object knowledge remain intact.[7]
- **Semantic variant**
  Patients manifest difficulties in naming and understanding everyday objects, particularly names of fruit, vegetables, and animals. Fluency and grammar remain preserved. At the beginning of the disease, less common words are replaced with more generic terms;[7] as the disease progresses, word comprehension worsens. Patients feel very challenged when describing images or objects to which they have to assign a name. Memory is not impaired; patients are able to perfectly remember recent events.[7]
  As time passes, they increasingly manifest: impaired object knowledge, surface dyslexia, and dysgraphia. Impaired object knowledge means that the concept of the object is lost, what the object is and what it is used for; surface dyslexia means that they also lose the pronunciation of objects; and dysgraphia means that they write words as they are pronounced.[7]
- **Logopenic variant**
  Patients present a non-fluent language, but intact grammar and comprehension. The main characteristic is difficulty in word retrieval, especially names of objects, both in spontaneous conversation and during naming exercises. Phonological errors are frequent, with sound exchanges or omissions. [7] This variant maintains comprehension and the meaning of individual words. They use grammatically correct language; the problem concerns errors or taking time to find the right word.[7]
- **Mixed variant**
  There is a group of patients who present simultaneously problems of grammar and comprehension from the early stages. They represent the mixed group.[16]This group is not included in the 2011 guidelines, published in the article Classification of Primary Progressive Aphasia and Its Variants.[1]

Figure 5 provides a summary of the main speech characteristics.

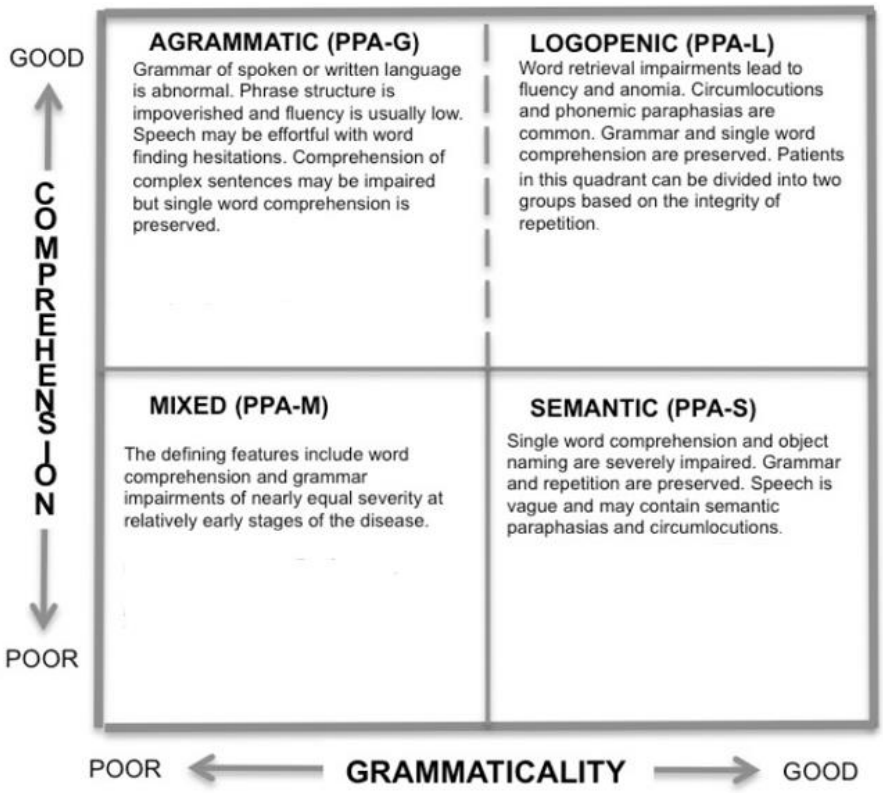

Figure 5: A 2-dimensional template for the classification of PPA patients. [16]

2. Magnetic Resonance Imaging

Magnetic resonance imaging (MRI) is the application of nuclear magnetic resonance (NMR) to radiological imaging. The word “magnetic” refers to the use of strong magnetic fields, while resonance means using an oscillating field at the same frequency as the precession of the nuclear spins in body tissues. [18]

Today, MRI is one of the most common medical imaging techniques, as it allows visualization of the inside of the human body in a non-invasive way.[19] Its main advantage is the absence of ionizing radiation, which makes the examination repeatable at short time intervals without risk to the patient. In addition, the method is highly versatile: it can produce three-dimensional images of internal structures and can also provide information on functional aspects of the body.[20] The main limitations are few: the presence of pacemakers or metallic implants may prevent the examination, while the equipment remains large, costly, and requires specially designed facilities. [19]

From a historical perspective, the foundations of magnetic resonance date back to 1946, when Bloch and Purcell first described the phenomenon of NMR, a discovery that earned them the Nobel Prize in the same year. [21] The idea of using this principle for diagnostic purposes was proposed in 1971 by Damadian, who demonstrated that the relaxation times T1 and T2 differed significantly between healthy and tumorous tissues. Just a few years later, in 1973, Lauterbur and later Mansfield and Grannell published the first studies using magnetic field gradients to obtain spatial information, laying the groundwork for modern imaging .[18]

The technique developed rapidly until 1980, when the first full-body clinical images were produced at the universities of Nottingham and Aberdeen, marking the beginning of the modern era of magnetic resonance imaging[21].

2.1 Basic principles of MR

2.1.1 Source of MR signal

All atomic nuclei are made up of protons and neutrons, but only some possess the property called nuclear spin *I*, which depends on the number of protons and neutrons in the nucleus.[21] Table 1 lists examples of nuclei with spin 0, ½, or 1, along with their main properties.

Spin	Relevant Isotopes	Common features of the nuclei/isotopes
0	^{12}C, ^{16}O	Nuclei composed of even numbers of protons and even numbers of neutrons
½	^{1}H, ^{13}C, ^{15}N, ^{31}P, ^{19}F, ^{129}Xe	Nuclei composed of odd number of nucleons (protons and neutrons)
1	^{2}H, ^{14}N	Nuclei composed of odd numbers of protons and odd numbers of neutrons

Table 1: Examples of nuclei with spin values of 0, ½, and 1 (the table does not include examples of spin values above 1) [22]

Nuclei with nonzero spin possess a magnetic moment μ_I , which causes them to behave like magnetic dipoles. These dipoles generate a small local magnetic field, similar to the one produced by a magnet with north and south poles(Figure 6).[21]

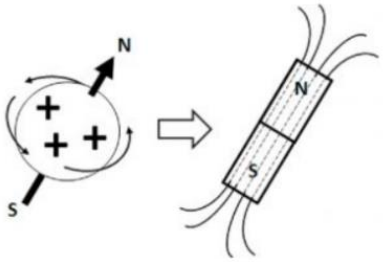

Figure 6: On the right, an atom with a magnetic moment μ_I that induces a magnetic field and behaves like a small bar magnet on the left. N and S indicates the north and south poles, respectively.[22]

More specifically, the magnetic moment μI is calculated using the formula:

$$\mu_I = I * \gamma * \frac{h}{2\pi} \quad (1)$$

Where I is the nuclear spin, γ is the gyromagnetic ratio of the nucleus, and h is Planck's constant, equal to $1.054 * 10^{-34}$ Js.

In MRI, the main signal comes from hydrogen, since more than 60% of the atoms in the human body are hydrogen. Imaging of different tissues depends on the fact that hydrogen atoms have different chemical and magnetic environments depending on the tissue type. [19]

Under normal conditions, nuclei are randomly distributed in space, with their magnetic moments μ_I oriented in random directions. This arrangement causes the magnetic moments to cancel each other out, so that the total magnetic moment of the tissue is effectively zero. [23]

When an external magnetic field B_0 is applied, nuclei with spin I can adopt different orientations according to the magnetic quantum number m_I, which ranges from $-I$ to $+I$ in integer steps, resulting in $2I + 1$ possible energy levels. For nuclei with spin $I = 1/2$, this corresponds to two distinct energy states:

- α state: $m_I = +1/2$, parallel alignment with the external field $B0$, lower energy.

- β state: $m_I = -1/2$, antiparallel alignment with the external field $B0$, higher energy.[23,24]

The energy difference between these two states, known as the Zeeman splitting, is determined by the interaction between the nuclear magnetic moment μ_I and the external magnetic field, and is given by:

$$\Delta E_I = \gamma * \frac{h}{2\pi} * B_0 \quad (2)$$

Where γ is the gyromagnetic ratio of the nucleus, h is the Planck constant and B_0 is the strength of the external magnetic field.[23] Figure 7 provides a visual representation of the energy splitting between the α and β states.

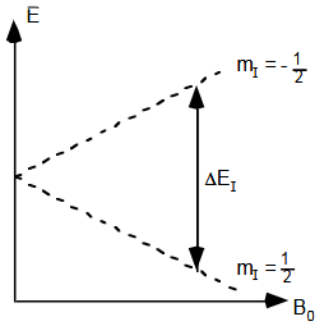

Figure 7: Difference of energy between state α and β. [19]

The imbalance in the population of nuclei between the two energy states is crucial for the formation of the magnetization vector Mz, which is parallel to the external field B_0 .[24]

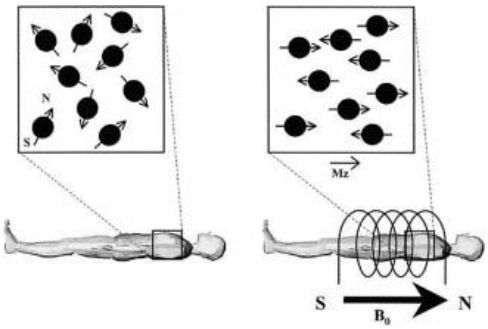

Figure 8: Differences between the magnetization vectors in the absence of an external magnetic field (B_0) and in the presence of an external magnetic field (B_0).[24]

The nuclei do not remain perfectly aligned along B_0, but undergo a motion called precession, that is, a circular movement around the direction of the external magnetic field. The frequency of this motion, known as the Larmor frequency, depends on the type of nucleus and on the strength of the magnetic field, according to the equation[24]:

$$f = B_0 * \frac{\gamma}{2\pi} \quad (3)$$

The precession phase differs for each nucleus, as shown in Figure 9.

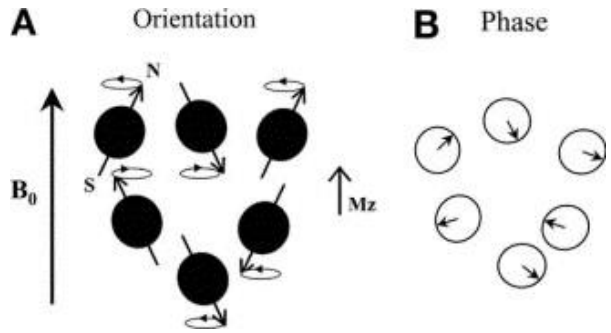

Figure 9: (A) Effect of nuclear precession, generated by the external field B_0, contributing to their orientation. (B) Precession phase around the axis of the external field, distinct for each nucleus. [24]

In commercial devices that use a B_0 field of 1.5 Tesla (T), the precession frequency for hydrogen is 63.75 MHz.[24]

2.1.2 Excitation phase

The net magnetization vector M_z, under equilibrium conditions, remains constant along the axis of the static magnetic field B_0 and does not produce a measurable signal .[24] In order to obtain a detectable signal, the orientation of the magnetization vector is altered by applying a radiofrequency (RF) magnetic field B_1, oriented perpendicularly to B_0 . The B1 field is generated at the resonance frequency, the Larmor frequency, and is delivered in the form of a short pulse with a duration on the order of microseconds.[21]

Following the RF pulse, protons absorb energy and transition from the lower-energy state (parallel to B_0) to the higher-energy state (antiparallel to B_0).[24] The energy required for this transition corresponds to the energy difference ΔE_I between the two states α and β, as described in the equation (2) .[21]

The RF pulse produces a 90° rotation of the magnetization vector M_z , displacing it from the positive z-axis into the transverse plane xy. In this plane, the transverse magnetization precesses around B_0 at the Larmor frequency. This precession generates a time-varying magnetic field that induces an alternating current (AC) in the receiving coil placed around the subject, allowing the detection of a measurable resonance signal, as shown in Figure 10. [24]

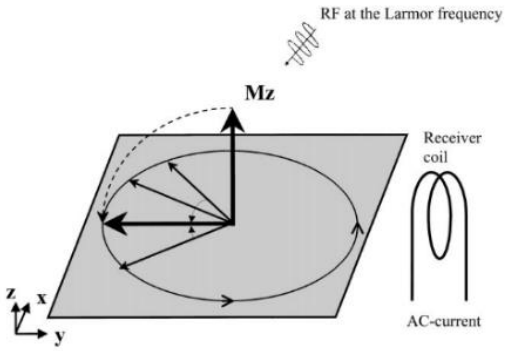

Figure 10: Excitation of spins with an RF pulse causes a 90° flip of the net magnetization Mz , inducing an AC current in the receiving coil.[24]

2.1.3 Relaxation phase

After RF pulse is turned off, the transverse magnetization M_{xy} decays over time, while the longitudinal magnetization M_z progressively increases, returning toward the equilibrium point. In this process, high-energy spins release energy and return to the low-energy state. The signal induced in the receiving coil decreases over time, a phenomenon known as Free Induction Decay(FID), shown in Figure 11. The time required for the magnetization to fully return to equilibrium is defined as the relaxation time.[24]

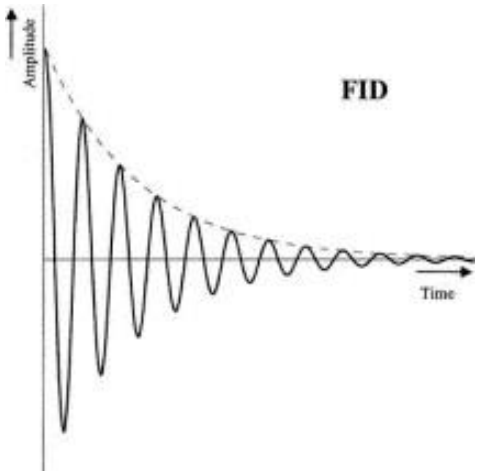

Figure 11: The FID signal shows a progressive decay over time, reflecting the return of the net magnetization vector to equilibrium.[24]

Relaxation is divided into two processes: longitudinal and transverse.

- **Longitudinal relaxation:** this process describes the realignment of magnetization with the external magnetic field. The spin-lattice relaxation time T_1 indicates how quickly the spins return parallel to the magnetic field B_0 along the z-axis; more precisely, it is the time required for the longitudinal magnetization to reach 63% of its equilibrium value.[24,25] The Bloch equation describing longitudinal magnetization is:

$$M_z(t) = M_0(\,1 - e^{-\frac{t}{T_1}}\,) \tag{4}$$

  where M_0 is the equilibrium magnetization, T_1 the relaxation time, and t the time elapsed since the RF pulse.[25]
  By substituting $t = 3 * T_1$ into the equation (4), M_z reaches approximately 95% of its equilibrium value.[25]

- **Transverse relaxation:** represents the decay of the transverse magnetization, caused by a dephasing among the precession frequencies of the spins.[24] This process is known as dephasing. [25] Immediately after the RF pulse, the spins are completely in phase; however, over time they begin to dephase due to spin-spin interactions between nuclei [24], assuming the absence of field

inhomogeneities.[25] This phenomenon is called transverse relaxation or spin–spin relaxation, and it is characterized by the time constant T_2. The time T_2 represents the interval required for the transverse signal to decay to 37% of its initial value.[24]

The formula describing the behavior of the transverse magnetization is:

$$M_{xy}(t) = M_{xy}(0) * e^{-\frac{t}{T_2}}) \quad (5)$$

where $M_{xy}(0)$ is the initial magnetization in the transverse xy-plane, t is the time elapsed since the RF pulse, and T_2 is the transverse relaxation time.[25]

The following Figure 12 shows the evolution of the transverse and longitudinal magnetization over time.

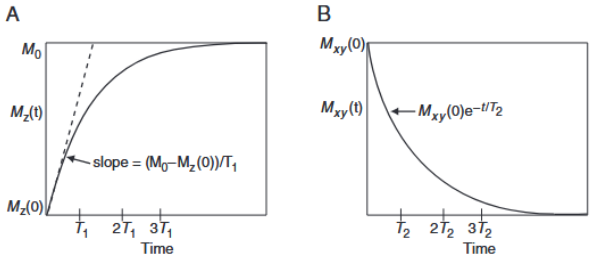

Figure 12: Visualization of the processes following the RF pulse: (A) Longitudinal relaxation, showing the recovery of the longitudinal magnetization from $M_z(0)$to the equilibrium value M_0. (B) Transverse relaxation, showing the decay of the transverse magnetization from $M_{xy}(0)$ to zero. [25]

The relaxation times T_1 and T_2 vary depending on the tissue, due to differences in molecular properties and microstructural environment. However, in all tissues, T_2 is always shorter than T_1, as shown in Table 2. [24]

Tissue	T_1 (msec)	T_2 (msec)
Gray matter (GM)	950	100
White matter (WM)	600	80
Muscle	900	50
Cerebrospinal fluid (CSF)	4500	2200
Fat	250	60
Blood	1200	100-200

[a]These are only approximate values. For the T_2 value of blood, the higher value pertains to arterial blood and the lower value to venous blood.

Table 2: Values of relaxation parameters T1 and T2, in msec, for different human body tissues composed by Hydrogen. Values are calculated at 1.5 T and 37°(Human body Temperature). [25]

In practice, inhomogeneities in the external magnetic field B0 and other local variations cause additional phase dispersion and, consequently, signal loss.[25] This apparent decay is referred to as T_2^*, which can be 2 to 50 times shorter than T_2. To

compensate for this effect, a subsequent 180° pulse is applied, capable of reversing the dephasing process and promoting signal rephasing. [24]

2.1.4 Pulse sequences

A pulse sequence defines how RF pulses are applied to generate detectable signals. Magnetic field gradients then allow each signal to be assigned a spatial location within the image.[26]

The FID signal, however, is not used directly to generate clinical images for several reasons. First, it decays before spatial encoding can be completed, even with ultra-fast MR scanners. Second, the use of pulse sequences allows the acquisition of images with controlled contrast, based on the T_1 and T_2 values of the tissues. [24] One commonly used sequence is the Spin Echo (SE) sequence, which consists of a 90° RF pulse that aligns the spins in the transverse plane but generates a phase dispersion in the precession frequency over time, followed by one or more 180° pulses that refocus the spins. [19]

After a 180° pulse, the magnetic vectors that precess at higher frequencies momentarily lag behind those with lower frequencies, but they ultimately re-align, giving rise to a signal peak referred to as the Spin Echo or Conventional Spin Echo (CSE).[19] The vectors rephase at a time $\frac{\text{Echo Time}(TE)}{2}$ after the 180° pulse, corresponding to the time TE after the 90° pulse. [19] The 180° pulse can be repeated multiple times, generating new echoes with decreasing amplitudes, which faithfully reproduce the true T_2.[19] The sequence is applied multiple times, typically 128–256 times, to uniquely encode the different spatial positions. [26]

The Spin Echo sequence, shown in Figure 13 has two key parameters:

- **Repetition Time($\boldsymbol{TR}$)**: the time interval between two consecutive 90° pulses, also referred to as the repetition time.
- ***TE***: the time between the 90° pulse and the detected echo signal in MRI, that is, the moment when the echo is formed. More precisely, the echo time is defined as the interval between the center of the 90° pulse and the center of the echo.[27]

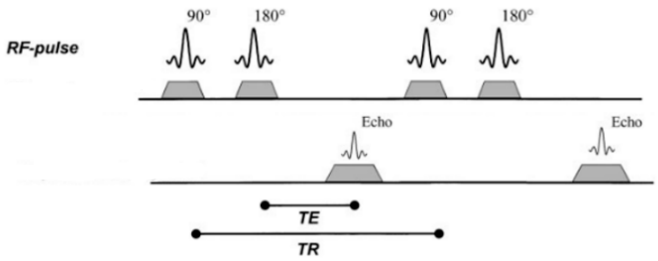

Figure 13: Example of a spin-echo sequence. An echo signal is evoked after a 90° and a 180° RF pulse. [24]

The choice of TE and TR determines whether the contrast of the resulting image is primarily based on differences in T_1, T_2, or the proton density of the tissues of interest. The resulting images are commonly referred to as T_1-weighted, T_2-weighted, and proton density-weighted, respectively. [26]

Other pulse sequences, such as Fast Spin Echo and Gradient Recalled Echo, have been developed to reduce scan time or provide different contrast mechanisms, but the Spin Echo sequence remains the fundamental reference for clinical imaging.[19]

2.1.5 Spatial Encoding

In MRI, the signal must carry information on the spatial location of hydrogen protons within the patient's body.[24]

Unlike other imaging modalities, such as X-rays, Computed Tomography(CT), ultrasound, or nuclear medicine, MRI receiver coils are not capable of collimating the signal and thus cannot directly determine its spatial origin.[28] To overcome this limitation, a spatial encoding scheme has been introduced.[24] This encoding is achieved by applying magnetic field gradients.[28] In particular, spatial information is obtained through three main steps:

1) **Slice selection** [24]: The first step consists in selecting the section to be analysed. For this purpose, the static and homogeneous magnetic field B_0 is perturbed by time-dependent magnetic field gradients.[28]

   A schematic representation of this process is shown in Figure 14.

   - Section a) illustrates a uniform magnetic field, in which the field vectors have the same length and thus the same intensity throughout the entire volume.
   - Section b) shows the application of a gradient field along the z-axis. In this case, the field intensity varies proportionally with the position along z, while remaining constant at any point in the x–y plane. This variation is also highlighted by a colour transition from red to blue, indicating the change in field intensity only along the z-axis. For any point within the x–y plane, the colour remains constant, showing the absence of variations in those directions.
   - Sections c) and d) illustrate, in an analogous manner, the application of a gradient in the x or y direction, producing a variation only along the selected axis, while leaving the other components unchanged.[28]

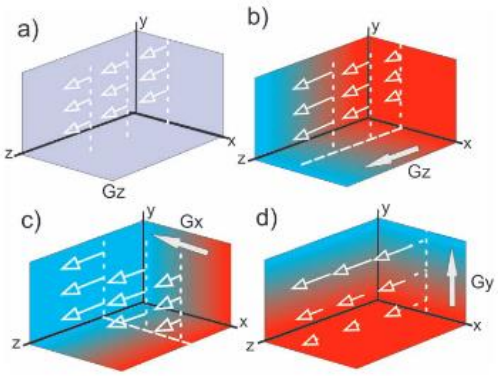

Figure 14: A uniform magnetic field (a) is modified by gradients Gz, Gx, and Gy, due to local field variations, which cause changes along the z, x, and y axes. [24]

By combining gradients along the three spatial directions, slices can be obtained in any orientation across the body. [24]

Gradients are not constant but have a magnitude that represents the rate at which the field changes per unit distance. A typical value ranges between 10 and 25 mT/m [24,28], which corresponds to a variation of millitesla per meter. Compared to the main homogeneous field B_0, the applied gradients represent only a minimal perturbation, on the order of 1%.[28] Thanks to this small perturbation, the Larmor frequency, which depends on the field intensity according to the equation (3), also becomes a function of position. By selecting a central excitation frequency for the RF pulse, it is possible to control the depth of analysis, while the slice thickness of interest is determined by selecting a frequency interval between f_1 and f_2.[24] Once the RF pulse is switched off, only the slice that has been excited within the selected frequency range generates a transverse magnetization vector, while the magnetization outside the slice remains aligned with B_0.[28] The most common slice thickness values range between 3 and 6 mm. [24] In this way, by adjusting the gradient direction, its intensity, and the frequency bandwidth, it is possible to control both the position and thickness of the selected slice(Figure 15).

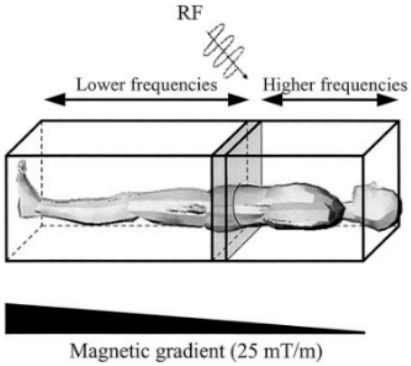

Figure 15: Illustration of the slice selection process. [24]

2) **Phase encoding** [24]: It is achieved by introducing a brief, temporary change in the magnetic field between the RF pulse and signal readout. This is done by applying a time varying gradient, commonly along the y-direction, G_y [29].
   During the application of this gradient, spins located at different positions along y precess at different frequencies. Once the gradient is switched off, the precession frequency becomes identical across the slice again, but each spin has accumulated a distinct phase shift proportional to its position along y. By repeating this process with varying gradient durations, signals with different phase encodings can be obtained.[24]

3) **Frequency encoding** [24] : It is used to determine the position of signals within the slice along the x-direction and to differentiate pixels that share the same phase encoding. A magnetic gradient G_x is applied during signal readout, causing the Larmor frequency to vary as a function of the x-position.[29]

By combining phase information (y-axis) and frequency information (x-axis), a two-dimensional grid is created, in which each pixel is defined by a unique combination of phase and frequency encodings. This grid is referred to as k-space.[24] In k-space, low spatial frequencies, which encode the overall image contrast, are located near the center, while high spatial frequencies, responsible for fine details, are positioned at the periphery, as shown in Figure 16. [30]

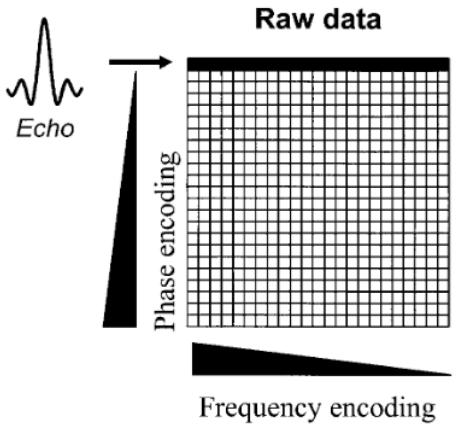

Figure 16: Frequency encoding, with a gradient during readout of the AC signal, is used to differentiate pixels with the same phase encoding. [24]

Subsequently, a Fast Fourier Transform (FFT) is applied along both the phase and frequency encoding directions. This operation converts the raw data, which represent signal amplitude as a function of time, into a domain where amplitude is expressed as a function of frequency. [30]
The acquisition time of an image is directly related to the number of phase-encoding steps required. For instance, in a 256 × 256 image, corresponding to 256 image lines, 256 distinct signals must be acquired.[24]

2.1.6 Contrast

In MRI, contrast is determined by the TR and TE times:

- $\boldsymbol{T_1}$-weighted images: Using a short TR and a short TE, produces T_1-weighted images, in which only protons with very short T_1 have enough time to recover their magnetization along the z-axis, while protons with long T_1 remain saturated and produce a weak or negligible signal.[19] In these images, tissues with short T_1, such as fat, appear bright, whereas tissues with long T_1, such as cysts, cerebrospinal fluid (CSF), and edema, appear dark.[26] **Errore. L'origine riferimento non è stata trovata.** shows an example of a T1-weighted image.

Figure 17: T_1- weighted image.[31]

- $\boldsymbol{T_2}$-weighted images: Using a long TR and a long echo time, highlights the differences in T_2.[19] In T_2-weighted images, tissues with long T_2 relaxation times, such as fluids, appear bright. In the brain, the differences in T_2 relaxation times between white matter and gray matter allow these tissues to be clearly distinguished.[26] **Errore. L'origine riferimento non è stata trovata.** shows an example of a T2-weighted image.

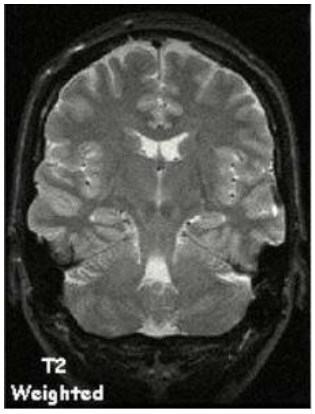

Figure 18: T_2- weighted image.[31]

- Proton density weighted images: Using a combination of long TR and short TE makes the contrast independent of both T_1 and T_2, depending almost

exclusively on the proton density of the tissue.[19] In this case, gray matter has a proton density approximately 20% higher than that of white matter, allowing a clear distinction between the two tissue types.[26] **Errore. L'origine riferimento non è stata trovata.** shows an example of a T1-weighted image.

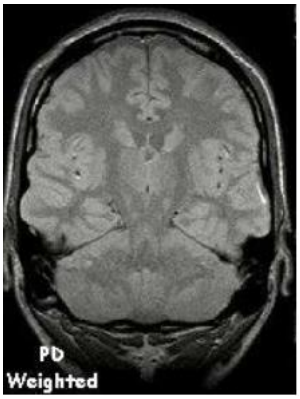

Figure 19: Proton density weighted image.[31]

The following Table 3 reports the typical TE and TR values used to achieve different image contrasts, expressed in milliseconds

IMAGE CONTRAST WEIGHTING	TE	TR
T_1-weighted	Short (15–25)	Short (500–700)
T_2-weighted	Long (80–100)	Long (2500–3500)
Proton density-weighted	Short (15–25)	Long (2500–3500)

Table 3: Typical TE and TR values. [26]

2.1.7 Resolution

The Field of View (FoV) in MRI represents the portion of the body that is captured in the image. It indicates the spatial dimensions of the area to be acquired in MRI and is determined by:

- Dimension along the x-axis: determined by the number of frequency encoding steps (e.g. 128 or 256).
- Dimension along the y-axis: determined by the number of phase encoding steps (e.g., 128 or 256).

For the same number of pixels, a larger FoV implies lower spatial resolution, as each voxel (volumetric pixel) represents a larger area. Conversely, a smaller FoV increases spatial resolution but reduces the area covered by the image.

The volume of each voxel is determined by the matrix size and the slice thickness. Spatial resolution depends on the pixel size: the smaller the pixels, the higher the resolution. However, a compromise is necessary: if the pixels are too small, they may not contain enough protons to generate a measurable signal, reducing image quality.[24]

2.2 Structural and Functional MRI Techniques

Magnetic resonance imaging is one of the most widely used neuroimaging techniques and can be divided into two main categories:

1. **Structural imaging:** refers to approaches specialized in the visualization and analysis of the anatomical properties of the brain. Structural approaches are particularly useful for studying brain damage and abnormalities, analyzing brain volume and cortical thickness. Classic structural magnetic resonance sequences include T1-, T2-, and proton density-weighted imaging.[32] Diffusion tensor imaging (DTI) is a variant of MRI that allows the study of brain connections by tracking the constrained movement of water molecules. The axis of this movement reflects the orientation of white matter tracts, i.e., the direction of connection tracts between different brain areas. This information can then be used to reconstruct the architecture of neural pathways along three dimensions, a technique known as tractography.[32] A more advanced structural technique is Voxel-Based Morphometry (VBM), which highlights focal gray matter atrophy.[33]

2. **Functional imaging:** refers to approaches based on the identification of regions of brain activity during the execution of specific cognitive tasks. A typical example is functional magnetic resonance imaging (fMRI), a variant of MRI that detects physiological changes in the brain related to blood flow and oxygen levels during task performance. This phenomenon is known as the blood oxygen level dependent (BOLD) response, and regional variations in this signal can be detected by the scanner.[32]

2.3 Applications of MRI in the Diagnosis of PPA

Structural MRI is the preferred imaging modality for the classification of PPA, due to its high soft tissue resolution and ability to accurately localize cortical atrophy. It enables the identification of specific atrophy patterns for each variant

- **NfvPPA**: Focusing on gray matter alterations, the most atrophied regions are the inferior frontal, opercular, and insular lobes of the dominant hemisphere, usually the left, often associated with a widening of the left Sylvian fissure.

The epicenters of the nfvPPA variant are the left inferior frontal gyrus and the opercular and triangular parts of the left frontal operculum.[33–35] This atrophy can be visualized in structural MRI, as shown in Figure 20.

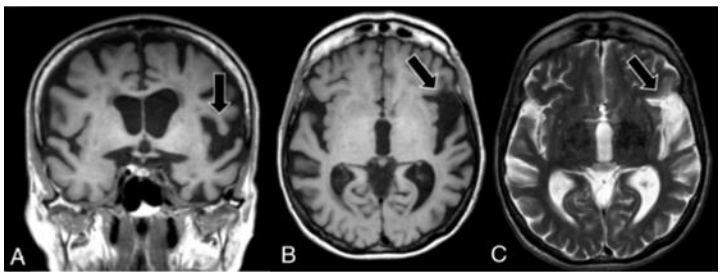

Figure 20: Coronal (A), axial (B) T1-weighted, and axial (C) T2-weighted MRI showing asymmetric widening of the left Sylvian fissure with predominant left posterior frontoinsular atrophy in a subject with nfvPPA. [33]

The greatest cortical thinning has been observed in the bilateral inferior frontal gyri, bilateral supplementary motor areas, the left temporoparietal junction, and the middle cingulate cortex. Additional affected regions include the supplementary motor area and the striatum, whose involvement has been associated with speech and articulatory impairments.[33]

As the disease progresses, atrophy extends to the right hemisphere and involves the posterior frontal regions, the supplementary motor area, the insula, the striatum, the inferior parietal regions, reaching the underlying white matter.[33]

Cortical thinning across both hemispheres can be appreciated in surface-based analyses, as illustrated in Figure 21.

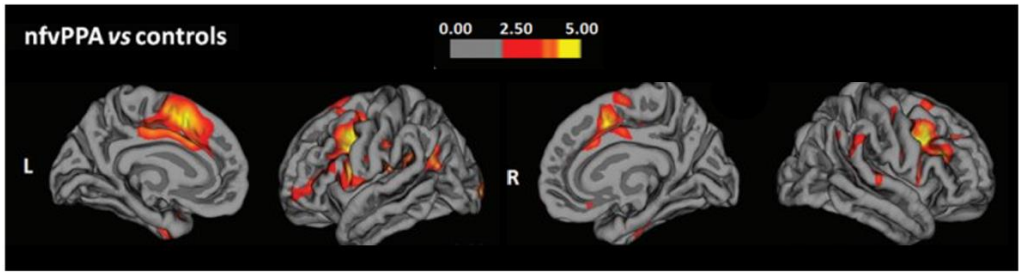

Figure 21: Cortical thinning on the pial surface in the left (L) and right (R) sides of the cortex in patients with nfvPPA.[36]

- **SvPPA**: The atrophy associated with the svPPA variant is regional, predominantly affecting the left temporal lobe, more pronounced anteriorly. In some patients, atrophy also occurs in the right temporal region.[33]

  At the cortical level, the greatest atrophy is observed in the bilateral fusiform and inferior temporal gyri, extending medially and affecting the amygdala, parahippocampal gyri, left temporal pole, middle temporal gyrus, and caudate nucleus. [37]

  This regional atrophy can be visualized using structural MRI, as shown in Figure 22.

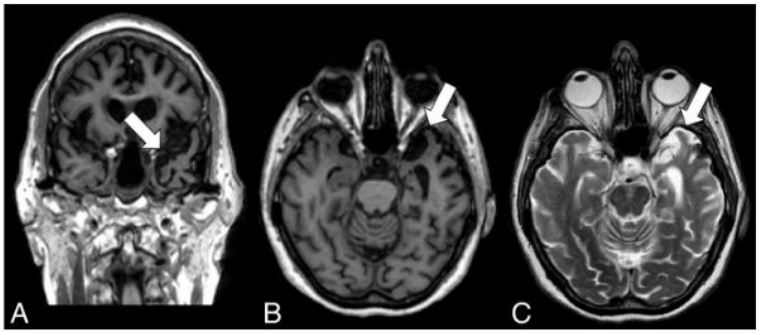

Figure 22: Coronal (A), axial (B) T1-weighted, and axial (C) T2-weighted MRI showing marked asymmetric atrophy of the left anterior temporal lobe in a subject with svPPA. [33]

At the surface level, there is marked cortical thinning in the left temporal lobe, particularly at the temporal pole, entorhinal cortex, parahippocampal, fusiform, and inferior temporal gyri, with a similar but less extensive involvement of the right hemisphere. .[33,38] Figure 23 illustrates this cortical thinning.

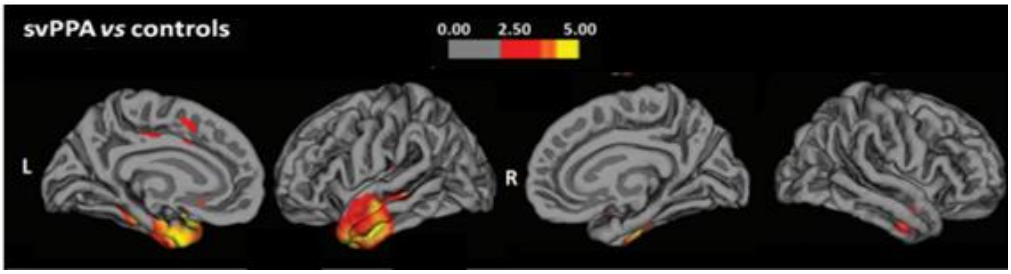

Figure 23: Cortical thinning on the pial surface in the left (L) and right (R) sides of the cortex in patients with svPPA. [36]

With disease progression, atrophy extends to involve the orbitofrontal cortex, frontal cingulate cortices, inferior and anterior insula, parietal operculum, precuneus, supramarginal and angular gyri, and superior parietal lobule.[39]

- **LvPPA**: This variant is characterized by atrophy in the posterior region of the left superior temporal gyrus, corresponding to Wernicke's area. In patients with PPA, an asymmetric widening of the Sylvian fissure can often be observed.[33] Additional affected regions include the left posterior superior temporal gyrus, the middle and superior temporal gyri, and the left inferior parietal gyrus. [40]The figure shows the atrophy pattern as seen on structural MRI.

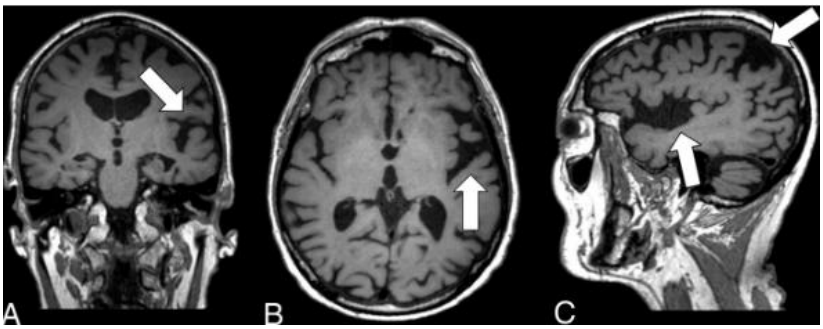

Figure 24: Coronal (A), axial (B), and sagittal (C) T1-weighted MRI showing asymmetric widening of the left Sylvian fissure with atrophy of the posterior peri-Sylvian region and left temporoparietal lobe in a subject with lvPPA.[33]

3. Artificial intelligence

Artificial intelligence (AI) emerged in the 1950s, when British mathematician Alan Turing proposed the idea of a machine capable of thinking. The formal birth of AI, however, is generally associated with Dartmouth College in New Hampshire, where a workshop brought together leading computer scientists of the time[41], including M. Minsky, A. Newell, H. Simon, N. Rochester, and C. Shannon[42], with the aim of developing a machine capable of simulating human learning, that is, an efficient system able to learn autonomously. John McCarthy, one of the conference organizers, coined the term "artificial intelligence." [41]

The AI field focuses on automating tasks traditionally performed by humans[43], using machine learning (ML) and deep learning (DL). ML is considered a subset of AI, while DL is a subset of ML.

3.1 Machine learning

3.1.1 Basic principles of ML

The term ML was introduced by Arthur Samuel in 1959 to describe "the ability of computers to learn without being explicitly programmed with new skills".[44] ML is a branch of artificial AI that focuses on developing models capable of learning from data and making predictions or decisions autonomously.[44,45]

Unlike classical programming, where an algorithm is explicitly coded using known features and the computer is provided with both a dataset and the algorithm specifying how to process the data to generate outputs(Figure 25 A), in ML the objective is to develop the algorithm itself. In this approach, the computer is given a dataset (combinations of features) along with the corresponding outputs, and the system learns an algorithm that captures the relationship between them (Figure 25 B).[43]

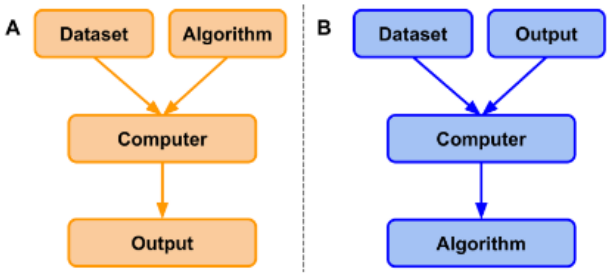

Figure 25: Classical programming (A) versus ML approach (B).[43]

ML is particularly useful for handling large volumes of data or solving complex problems that would require a long time for humans to process.[45]

The ML process can be schematized as a cycle comprising the following phases:[46]

1) **Data management**: This phase involves the collection and preparation of data, which may include sounds, texts, numerical or categorical values, or images. [44] Before being used for training, data typically undergo three main steps:
   - Encoding and Preprocessing: Categorical data are converted into numerical values, since most algorithms require numerical input .[47]If the dataset is limited, data augmentation techniques may be applied to increase its size. Additionally, noise is removed to ensure the dataset is consistent, and normalization or standardization can be performed to prepare the data for training.[46]
   - Feature extraction: Original data are transformed into a new set of features designed to capture the most relevant aspects of the data and reduce its complexity.[48]
   - Feature selection: A subset of features is selected to reduce model complexity, remove irrelevant characteristics, and allow for more efficient training of ML algorithms.[49]

   During this phase, the dataset is also divided into a training set, validation set, and test set.
2) **Model training**: The choice of model depends on the type of problem to be solved.[46] The model learns the relationships between inputs and outputs using the training set.[44] A loss function is then defined to measure the model's error. The objective of the training phase is to produce a model that minimizes this error through parameter optimization and hyperparameter selection.[46,50] This step also involves the use of the validation set, which aims to identify and prevent overfitting, a phenomenon in which the model memorizes the training data instead of generalizing properly.
3) **Model testing**: The model is evaluated using the test set, consisting of data not seen during training and validation. In this phase, the model's ability to generalize to new data is assessed, and the main performance metrics are calculated.[46]
4) **Update and maintenance**: The model can be periodically retrained or adapted in the presence of new data, in order to maintain optimal performance over time.[46]

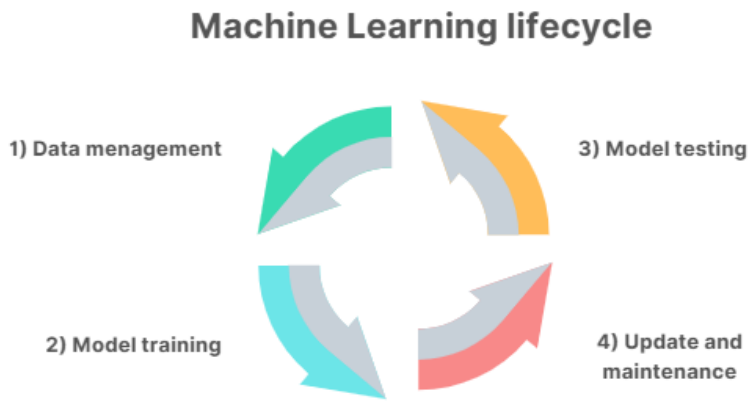

Figure 26: Iterative cycle of ML.

The main types of ML are: supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning.

1) Supervised learning:

This type of learning is based on building a model trained on a labeled dataset $\{(x_i, y_i)\}_{i=1}^{N}$, where:

- x_i represents the feature vector that describes the N input example. Each dimension $j = 1..D$ of the vector contains a value, called a feature, which describes some aspect of the input example. In this way, each example is represented by a set of features that characterize its properties.[51]
- y_i is the label associated with the example, which can take different forms:
  - An element belonging to a finite set of classes.
  - A real number.
  - A more complex structure (e.g., a vector or a matrix).[51]

The goal of supervised learning is to build a model that, starting from the features extracted from the examples, can produce the correct output for new, unseen data. [51]

Supervised learning can be divided into two main categories:[44]

- Classification is the task of predicting the category or class of an input. It can be either binary (two classes) or multiclass (more than two classes).
- Regression is the task of predicting continuous numerical values from input data.[43,44]

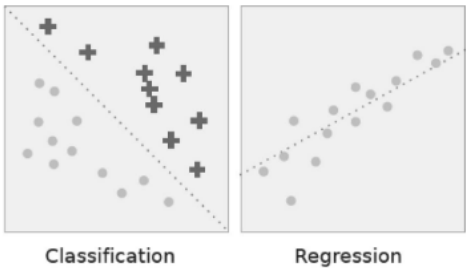

Figure 27: Classification vs. regression. In classification, the dotted line indicates a linear boundary dividing the two classes, whereas in regression, it represents the linear relationship between the two variables. [49]

2) Unsupervised Learning:

In contrast to supervised learning, in unsupervised learning the dataset consists of unlabeled data $\{(x_i)\}_{i=1}^{N}$. Here, x_i also represents a feature vector.[51]

The goal of an unsupervised algorithm is to process an input feature vector and produce as output:

- Another vector
- A value that can be used to solve a practical problem.[51]

Based on the extracted features, the algorithm can identify patterns within the dataset and group the data into categories. It is called "unsupervised" because the

patterns are determined directly by the algorithm, without any predefined correct answer. In this way, the algorithm learns to detect patterns and cluster the data autonomously.[43]

Unsupervised learning can be mainly divided into:

- Density estimation: estimates the distribution of the data.[44]
- Clustering: groups data into clusters based on similarities and differences.[44]
- Dimensionality reduction: produces a feature vector with fewer dimensions than the input.[51]
- Outlier detection: outputs a real number indicating how much a given example deviates from a "typical" instance in the dataset.[51]
- Generation: generates new data similar to the original dataset.[44]

3) Semi-supervised Learning:

The dataset contains both labeled and unlabeled examples, and typically the number of unlabeled examples is much larger than the labeled ones. The goal of a semi-supervised algorithm is the same as that of a supervised algorithm: to build a model capable of predicting the labels of the examples. Using a large number of unlabeled examples can help the algorithm produce a better model.[51]

4) Reinforcement Learning:

This type of learning lies between supervised and unsupervised learning. Unlike unsupervised learning, there is a form of supervision, but it does not correspond to a known output for each input in the dataset. It is an algorithm that receives feedback from the environment only after selecting an output for a given input or observation. The feedback indicates how well the output, called an action, contributes to achieving the learning objectives. Reinforcement learning is used in sequential decision-making problems, where a sequence of actions (outputs) must be chosen based on observations (inputs), receiving feedback for each action. The optimal action is the one that maximizes the objective.[52]

3.1.2 Classical Methods of ML

In this section, the main ML algorithms for supervised and unsupervised learning are presented. The most common methods for supervised ML are:

1) Linear Regression

Linear regression identifies the relationship between one or more numerical features and a single numerical target value. It is an analysis technique in which the dataset is described using a straight line. [43]

2) Logistic Regression

Logistic regression is a classification algorithm whose goal is to find a relationship between features and the probability of a particular outcome.

Unlike linear regression, which uses a straight line to estimate the target value, logistic regression uses a sigmoid curve to estimate the probability of belonging to a class.

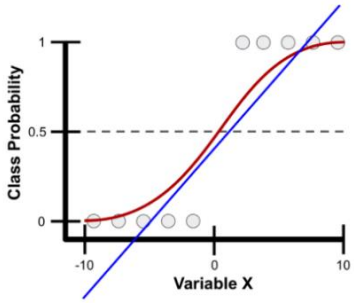

Figure 28: Example of Class Probability Prediction: Linear (blue) and logistic (red) regression models estimate the probability that samples (gray circles) belong to a class based on the variable X (-10 to 10).Logistic regression transforms X into probabilities between 0 and 1 using the sigmoid function, while linear regression directly assigns the class, choosing 0 or 1 based on the prevailing probability. [43]

3) K-Nearest Neighbors

The k-Nearest Neighbors (k-NN) method is an algorithm that predicts the class of data based on similarity measures.[49] It calculates the distance between the test data and all points in the training set and assigns the class through a simple majority vote among the k nearest neighbors.[44]

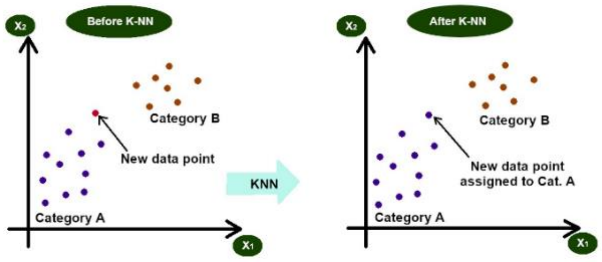

Figure 29: Example of classifying a new data point using k-NN. [44]

4) Decision tree

The Decision Tree (DT) is a method used for both classification and regression. Its operation is based on a tree structure. The classification of instances is performed by traversing the tree from the root to one of the leaf nodes. At each node, the corresponding attribute is evaluated, and the traversal follows the branch that corresponds to the attribute value, until the final decision is reached.[49]

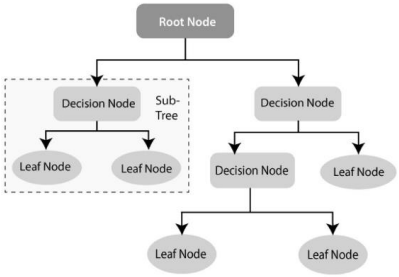

Figure 30: An example of a decision tree structure. [49]

5) Native Bayes Classifier

The Naive Bayes classifier is a probabilistic algorithm based on Bayes' theorem that assumes the independence of variables.[44] New data are classified by estimating the probability of belonging to a given class.[49]

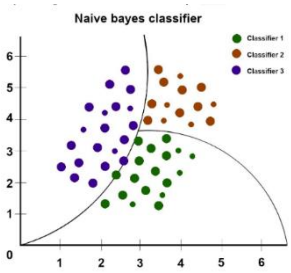

Figure 31 A graph depicting the performance of a Naive Bayes classifier.[44]

6) Support Vector Machine

A Support Vector Machine (SVM) bases its classification on finding the hyperplane that best separates objects belonging to different classes, maximizing the distance (margin) between the closest training points, called support vectors, in order to reduce the error. [44,49]

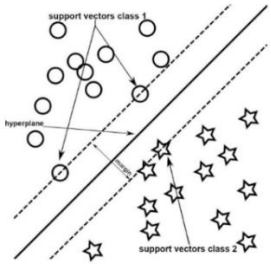

Figure 32: n example of an SVM classifying two classes. [44]

The most common methods for unsupervised ML is:

1) K-means clustering

The K-means is an algorithm that provides reliable results when the data are well separated.[49] Its purpose is to divide N data points into K clusters. [53] The algorithm consists of two main phases:

1. Initialization phase, in which each data point is assigned to the nearest centroid (the centres of the clusters).

2. Centroid update phase, in which the centroids of the clusters are updated by calculating the mean of the points belonging to each cluster.[53]

The iterative process ends when no data point changes cluster or the maximum number of iterations is reached.[53]

3.1.3 Optimization and Regularization

Every ML algorithm improves its performance during training thanks to a method that allows it to adapt to new data and increase its accuracy. Learning methods can be classified based on how the algorithm learns:

- **Evolutionary Algorithms (EA):** These algorithms learn iteratively, based on evolutionary principles. The model that solves the problem, called the genotype, is characterized by a set of properties. The performance of the model is evaluated using a fitness function. After calculating the fitness scores of the models, the next iteration generates a new model by applying mutations and crossover to the previous models that produced the most accurate estimates.

- **Algorithms based on Stochastic Gradient Descent (SGD):** These aim to minimize a loss function defined over the model outputs by adjusting the model parameters in the direction of the negative gradient. The gradient is calculated on a subset of the data (the training data), which is why the descent is called stochastic. The loss function represents the error that the model must minimize. The training process generally follows these steps:
    1. Present a batch of randomly sampled training data.
    2. Calculate the loss function between the model outputs and the desired outputs.
    3. Compute the gradient with respect to the model parameters.
    4. Update the model parameters in the direction of the negative gradient, multiplied by a chosen learning rate.
    5. Repeat until convergence.[54]

The performance of algorithms also depends on the choice of hyperparameters. For example, in the case of SGD, it is necessary to define the batch size, learning rate, and model initialization; in EAs, relevant parameters include the fitness function, population size, and crossover and mutation rates. There are also algorithms capable of automatically optimizing hyperparameters, thereby improving model performance regardless of the chosen learning method.[54]

Subsequently, each ML algorithm, after the training phase, must demonstrate its generalization ability on new, unseen data (test set). This ability determines how well a model can apply the learned patterns to make accurate predictions not only on the training data but also on new data. [55] The generalization error arises from this concept

and corresponds to the misclassification rate calculated on data that the model has never encountered before.[56]
Conversely, poor generalization leads to overfitting and underfitting phenomena, which cause inaccurate predictions.

- **Overfitting** occurs when the model learns too many details from the training data, becoming ineffective on new data. This happens when the model is too complex, the dataset is too small, or the training is too long. In this case, the model specializes in the training data instead of learning general rules.[55,57]
- **Underfitting** occurs when the model is too simple to learn meaningful patterns from the data. The model shows poor performance on both training and test data. [55,57]

There are several techniques to improve generalization ability in ML:

- **Cross-validation:** a technique that evaluates the robustness of the model by dividing the dataset into k subsets, called folds. The model is trained on some folds and tested on the remaining ones, reducing dependence on a single dataset and providing a more reliable estimate of performance. [55,57]
- **Regularization:** imposes constraints on the model's complexity to limit overfitting. The most common techniques are L1 and L2, which control complexity and prevent over-adaptation to data.
    - L1 (Lasso): eliminates less important features.
    - L2 (Ridge): penalizes large coefficients, distributing weights more evenly.[55,57]
- **Data augmentation:** increases the amount of training data by introducing variations of the original data, such as rotations, zooming, or noise addition, providing more examples to the model and improving its ability to generalize. [55,57]

3.1.4 Evaluation and Protocols

As mentioned earlier, to train a model it is necessary to divide the data into distinct sets: the training set, used exclusively to train the model; the validation set, used to determine the optimal parameters that minimize the validation error; and the test set, consisting of data never seen by the model, used to evaluate its final performance.[58]
At the data splitting level, there are different approaches to creating these initial splits:

- **Random split:** The data are divided randomly, reserving a larger portion for the training set and smaller portions for the test sets. The training set is further split into training and validation sets. This process is usually repeated multiple times, and the final estimate of the model's performance is obtained by averaging the results on the validation sets across all repetitions.[58]
- **Cross-validation (k-fold CV):** In this approach, the test set is kept separate. The training set is divided into k equally sized subsets, called folds. Of these

subsets, k-1 folds are used to train the model, while the remaining fold serves as the validation set. The process is repeated k times, so that each fold is used once as validation. The final performance is then evaluated as the average across all folds.[56,58]

- **Leave-One-Out Cross-Validation (LOO-CV):** This is an extreme variant of k-fold cross-validation, in which the number of folds corresponds to the total number of samples in the dataset. In this case, the validation set consists of a single sample. This technique is primarily suitable for very small datasets.[59]
- **Leave- Subject-out Cross-Validation(LSO-CV):** This is a variant of k-fold cross-validation in which the folds are defined by subjects rather than individual samples. The model is evaluated on subjects not included in the training folds. When each fold corresponds to a single subject, the method is referred to as Leave-One-Subject-Out Cross-Validation (LOSO-CV).[60]
- **Nested Cross-Validation:** Nested cross-validation is used when one aims to optimize a model's hyperparameters while simultaneously obtaining a more reliable estimate of its performance. The procedure involves two levels of cross-validation. In the first level, known as the outer loop, the dataset is divided into k folds, each of which serves in turn as a test set to evaluate the model's final performance. In the remaining folds, a second level of cross-validation, the inner loop, is applied to train the model and validate different hyperparameter configurations. In this way, the selection of hyperparameters occurs exclusively on the internal training and validation data, while the outer folds remain independent and are used solely to estimate the overall performance of the model.[61,62]

Whenever model performance is mentioned, it refers to the ability of the model to generalize to unseen data. For the evaluation of model performance, in regression problems the Mean Square Error(MSE) is typically used, which allows comparing the model's performance on the training set and the test set.[63]

In classification problems, both binary and multiclass, the most used method is to calculate performance metrics derived from the confusion matrix (CM).[63] The CM is a square matrix of size NxN, where N denotes the number of output classes of the model. Each row represents the number of instances of the predicted class, while each column represents the number of instances of the actual class.[64] Therefore, the matrix provides a detailed analysis of the classifier's performance. Since the true class labels are required, CM can only be used in supervised learning methods.[64]An illustration of the confusion matrix is shown in Figure 33.

In binary classification, predictions can be correct or incorrect, and the target labels can be positive or negative. The main terms are:

- **True Positive (TP)**: both the predicted and actual labels are positive.
- **False Positive (FP)**: the prediction is positive, but the actual label is negative (Type I error).

- **True Negative (TN)**: both the predicted and actual labels are negative.
- **False Negative (FN)**: the prediction is negative, but the actual label is positive (Type II error). [64]

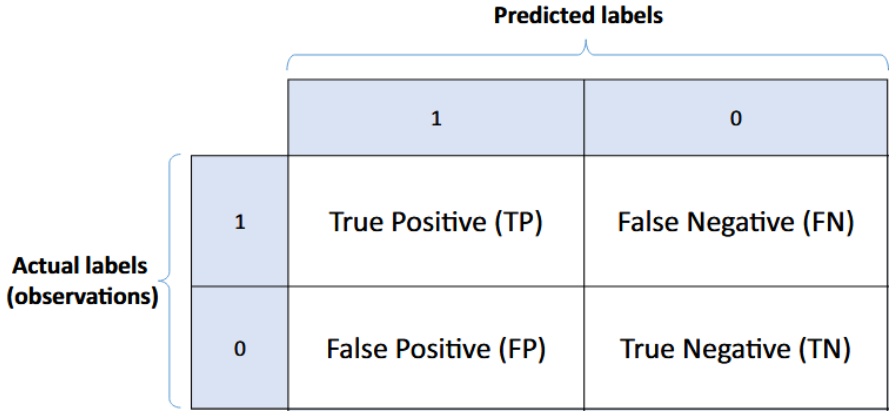

Figure 33: The confusion matrix for a two-class problem shows how well the algorithm predicted labels 0 ("negative") and 1 ("positive") by comparing the predicted and actual values. [64]

From these four parameters, the main performance metrics can be calculated:

- Sensitivity or Recall: the ratio of correctly predicted positive values to the total number of positive values in the dataset.

$$Sensitivity = \frac{TP}{TP+FN} \quad (6)$$

- Specificity: the ratio of correctly predicted negative values to the total number of negative values in the dataset.

$$Specificity = \frac{TN}{TN+FP} \quad (7)$$

- Positive Predicted Value (PPV) or Precision: the ratio of correctly predicted positive values to the total number of predicted positive values.

$$PPV = \frac{TP}{TP+FP} \quad (8)$$

- Negative Predicted Value (NPV): the ratio of correctly predicted negative values to the total number of predicted negative values.

$$NPV = \frac{TN}{TN+FN} \quad (9)$$

- Accuracy: the number of correctly predicted values divided by the total number of predicted values.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (10)$$

- Balanced Accuracy (BA): the average of sensitivity and specificity, useful in case of class imbalance.

$$Balanced\ Accuracy = \frac{1}{2}\left(\frac{TP}{TP+FN} + \frac{TN}{TN+FP}\right) \quad (11)$$

- F1-score : the harmonic mean of precision and recall, useful for imbalanced datasets.

$$F1 - score = 2\left(\frac{Precision\ x\ Recall}{Precision+Recall}\right) \quad (12)$$

3.2 Deep learning

Deep Learning is a subfield of ML, and more broadly of Artificial Intelligence. The term DL was first introduced by Hinton et al. in 2006. The development of DL triggered a resurgence in neural network research, to the point that it has sometimes been referred to as "next-generation neural networks." DL uses multiple hidden layers to represent data abstractions and build computational models. The term "Deep" refers precisely to the multiple levels or stages through which data are processed. Although DL models have been successfully applied in various fields, building an appropriate model remains complex. Furthermore, DL models are often considered "black boxes", which limits a complete understanding of their internal processes and the standard development of research and applications.[65]

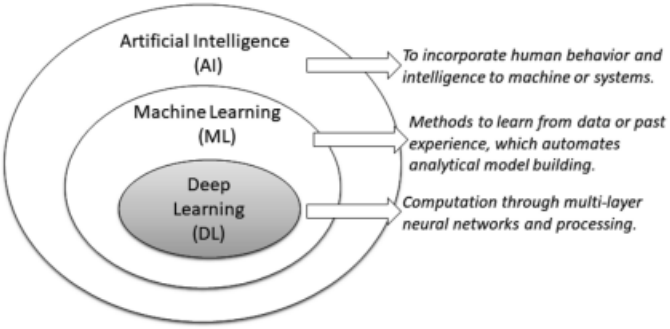

Figure 34: An illustration of the position of DL, comparing with ML and AI.[65]

3.2.1 Artificial Neural Networks

Artificial Neural Networks (ANNs) represent a subset of ML and constitute the foundation of DL techniques. Their name and architecture are inspired by the human brain, providing a simplified representation of how biological neurons function.[66]
Neurons communicate through electrical signals that originate in the dendrites, travel through the cell body and along the axon, and reach the synaptic terminals, where the signal is transmitted to other neurons.[67] Figure 35 illustrates the anatomical structure of a neuron.

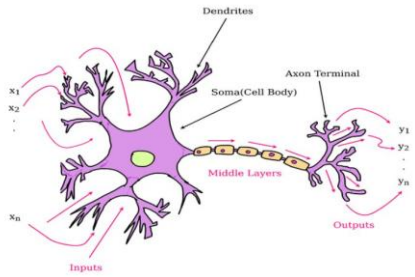

Figure 35: Anatomy of a biological neuron showing input and output structures. [68]

From a physiological perspective, neurons receive input signals and generate an output only when these signals exceed a certain threshold. The same principle is replicated in ANNs through the introduction of an activation function, which modulates the output as a function of the input.[69]

Similar to biological neurons that receive multiple presynaptic inputs and sum them, the neurons of an ANN also combine their inputs before generating an output. The sum of the inputs is then passed through an activation function, and the result corresponds to the neuron's output.[69]

A single neuron, also called a perceptron, is shown in the Figure 36 and its behavior can be mathematically described as follow:

$$y = f\left(\sum_{i=1}^{D} w_i x_i + b\right) \tag{13}$$

where D denotes the dimension of the input space, x represents the input vector, w is the set of weights corresponding to the input vector, b denotes the bias term, and f is the activation function.[70]

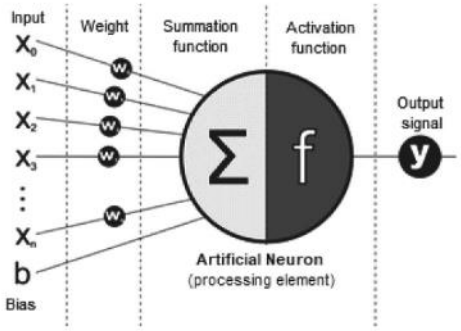

Figure 36:Graphical representation of a perception.[65]

Since biological neurons are interconnected, ANNs are similarly organized into layers of interconnected node, as illustrated in Figure 37.

An ANN is typically structured into the following layers:

1. Input layer: composed of artificial neurons that receive the initial data and transmit it to the subsequent hidden layer. This layer initiates the the network's processing flow.[66]
2. Hidden layer: consists of neurons that receive input from the previous layer and produce output toward the next layer. The inputs and outputs of these neurons are modulated by weights, which determine the strength of the transmitted signal. [66]High weights amplify the signal, while low weights attenuate it. [69]
   ANNs are characterized by the presence of a single hidden layer.[44]
3. Output layer: composed of the network's final neurons, which provide the results according to the specific task.[66]

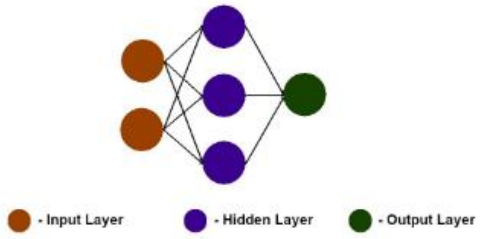

Figure 37: Graphical representation of an artificial neural network (ANN)[44]

Going into more detail regarding the activation functions mentioned earlier, the most commonly used include:

- Sigmoid (or logistic) function:
  This function, illustrated in Figure 38 and expressed in equation 14, compresses the output into the interval [0, 1], so is therefore not zero-centered. For very large or very small input values, it tends to saturate, causing the vanishing gradient problem: the gradient of the objective function with respect to the parameters becomes almost zero, preventing significant updates during training with gradient descent. As a result, training can slows down drastically or even get stuck. Moreover, the fact that the output is not zero-centered can reduce the convergence speed.[67]

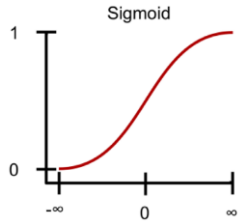

Figure 38: Graph of the logistic sigmoid function. [43]

$$Sigmoid(x) = \frac{1}{1 + e^{-x}} \qquad (14)$$

- Hyperbolic tangent function (Tanh):
  This function, illustrated in Figure 39 and expressed in equation 15, is zero-centered and compresses the input into the interval [-1, 1]. However, it shares the same limitations as the sigmoid function, including the vanishing gradient problem and higher computational complexity.[67]

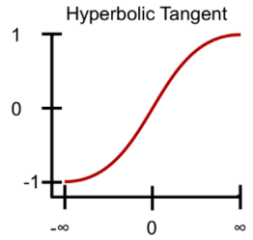

Figure 39: Graph of the hyperbolic tangent function. [43]

$$Tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \qquad (15)$$

- Rectified Linear Unit (ReLU) function:
  For positive inputs, it behaves like the identity function, while for negative inputs it outputs zero. The output range is [0, ∞). ReLU addresses the computational complexity issues of Sigmoid and Tanh, and despite the vanishing gradient for negative inputs, it is currently the most widely used activation function in state of the art models. [67] This function, is illustrated in Figure 40 and expressed in equation 16.

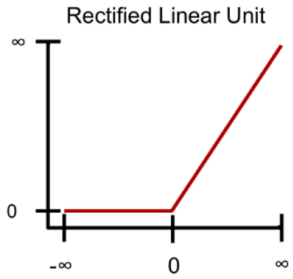

Figure 40: Graph of the rectified linear unit function. [43]

$$ReLU(x) = \max(0, x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases} \qquad (16)$$

- Softmax:

This function, illustrated in Figure 41 and expressed in equation 17 is typically used in the output layer of classification networks to transform a vector of real numbers into a probability distribution, which facilitates class assignment. It is particularly used in multiclass classification problems. [72]

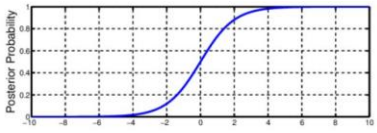

Figure 41: Graph of the softmax function.[71]

$$p(x_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad (17)$$

- z_i: the i-th input value
- K: the size of the input vector (the number of classes).
- $p(x_i)$the probability assigned to class i.

3.2.2 Multilayer Perceptron

The Multilayer Perceptron (MLP) is the simplest and most well-known neural network. It has the same architecture as an ANN, but it has multiple layers of neurons connected in parallel. The neurons in one layer do not communicate with each other but send information to the next layer. This unidirectional flow, from inputs to outputs, is called feedforward. MLPs are composed of an input layer, one or more hidden layers, and an output layer. The hidden layers map the input signal into another space in a non-linear way, through activation functions. The output layer has a number of neurons that varies depending on the desired output of the system. In regression or binary classification problems, there is only one output node. In the presence of multiclass classification, one output node is used for each class. To assign the class, an activation function is used in the output node. If the output value is greater than 0.5, one class is predicted; if the output value is less than or equal to 0.5, the other class is predicted. In the multiclass case, the class with the highest value is assigned.[73]

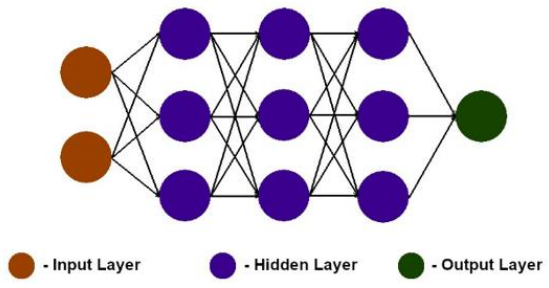

Figure 42: Graphical representation of a MLP. [44]

3.2.3 Learning and optimization in DL

The training of an MLP network is usually supervised, in which both the input variable values and the desired output values are provided as input.[73]
The training consists of two phases:

1. Forward step: In this phase, the network weights are fixed and are not updated. The input data are propagated from the first layer to the last layer, and subsequently an error function is calculated by comparing the obtained output with the desired output. [73]
2. Backward step: Starting from the loss function, the gradient is computed and, using a gradient descent-based optimization method, the weights of the neurons are updated in the opposite direction of the gradient in order to minimize the loss function.[73]
   The network starts from arbitrary initial weight values and, iteratively, approaches the minimum of the loss function, as illustrated in the Figure 43. [69]

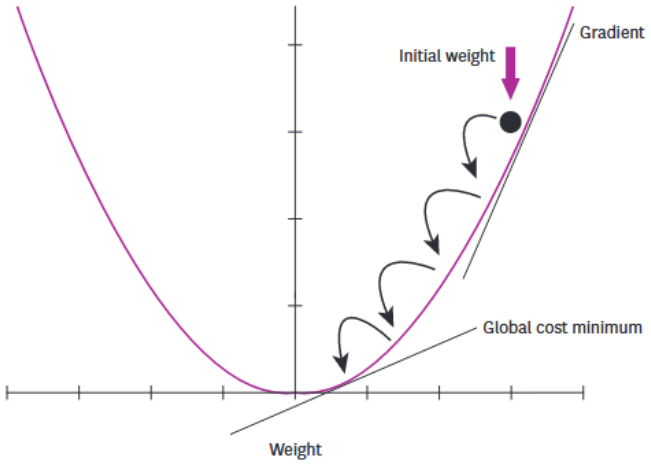

Figure 43: Illustration of weight updates during the backward step. [69]

These two steps are repeated iteratively until a stopping criterion is reached. Each complete iteration is called an epoch.[73]
Going into more detail, the loss functions vary depending on whether the tasks involve regression and classification. The most common ones are presented below.

For regression problems:

- Mean Squared Error:
  MSE is commonly used in regression problems and measures the average of the squared differences between the predicted and the true values:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \quad (18)$$

where n is the number of samples, is y_i the true value, and $\hat{y}_i$ is the predicted value of the $i-th$ sample.[74]

- Mean Absolute Error (MAE):
  MAE, also used in regression problems, measures the average of the absolute differences between the predicted and the true values:[74]

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{19}$$

- Root Mean Squared Error (RMSE):
  RMSE is the square root of the MSE:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{20}$$

  The RMSE measures the average deviation of the predictions from the true values. This metric is easy to interpret because it is in the same units as the data. However, it is sensitive to outliers.[74]

For classification problems:

- Binary Cross-Entropy Loss (BCE):
  BCE, also known as log loss, is a loss function commonly used for binary classification problems. It measures the dissimilarity between the predicted probability of a class and the true class label. The predicted probability is represented as a vector of probabilities for each class, where the predicted probability of the true class is $p(y = 1|x)$ and that of the other class is $p(y = 0|x)$. The loss function is defined as:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{21}$$

  Intuitively, it can be divided into two cases:
  $-\log(p) \quad if\ y = 1$
  $-\log(1-p) \quad if\ y = 0$
  where y is the true class label (0 or 1) and p is the predicted probability of the positive class.[74]

- Weighted Binary Cross-Entropy (WBCE):

WBCE is a variant of the standard BCE, in which the weight of each sample is considered in the loss calculation. This is useful in situations with class imbalance. The WBCE computes the loss as:

$$L = -(w_i \cdot ylog(p) + w_i \cdot (1 - y)\log(1 - p)) \quad (22)$$

where w_i is the weight assigned to sample i, y is the true label, and p is the predicted probability of the positive class. Assigning a higher weight to samples from under-represented classes balances their influence during optimization.[74]

- Categorical Cross-Entropy Loss (CCE)
  CCE also known as negative log-likelihood loss or multi-class log loss, is used for multi-class classification problems. It measures the dissimilarity between the predicted probability distribution and the true distribution.
  Given the predicted probability distribution, the loss is defined as the average negative log-likelihood of the true class:

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{i,j}\log(p_{i,j}) \quad (23)$$

  where N is the number of samples, C is the number of classes, y is the true label, and p is the predicted probability of the true class.[74]
  A weighted version of cross-entropy also exists for multiclass problems, to be used in cases of class imbalance.

Next, by analyzing the gradient of the loss function, the three variants of gradient descent are introduced. The loss function is denoted as $J(\theta)$, parameterized by the model weights $\theta \in \mathbb{R}^d$, while the gradient is $\nabla_\theta J(\theta)$. The symbol ηrepresents the learning rate, which determines the size of the steps taken to reach a (local) minimum. The three variants differ depending on the amount of data used. [75]

- **Batch Gradient Descent**
  Batch gradient descent calculates the gradient of the loss function with respect to the weights θ on the entire training dataset:

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta) \quad (24)$$

  Gradient descent is very slow, since it requires the entire dataset to compute the gradient. Moreover, this method is applicable only if the entire dataset can be loaded into memory.
  Batch gradient descent is guaranteed to converge to the global minimum for convex error surfaces and to a local minimum for non-convex surfaces. [75]
- **Stocastic Gradient Descent**

SGD performs a parameter update for each training example $x^{(i)}$and its corresponding label $y^{(i)}$:

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x^{(i)}; y^{(i)}) \quad (25)$$

This method is generally much faster than batch gradient descent and can also be used for online learning, as it performs frequent updates with high variance, causing the objective function to fluctuate significantly.
These fluctuations, on one hand, allow SGD to jump to new and potentially better local minima; on the other hand, they make it more difficult to converge to the exact minimum, since SGD may continue to overshoot it. [75]

- **Mini-batch Gradient Descent**
  Mini-batch gradient descent combines the advantages of the previous techniques and updates the parameters for each mini-batch of n training examples:

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x^{(i:i+n)}; y^{(i:i+n)}) \quad (26)$$

This technique helps reduce the variance of weight updates, promoting more stable convergence, and allows the use of highly efficient matrix optimizations, making the computation of the gradient for each mini-batch faster. Common mini-batch sizes range from 50 to 256. [75]

Subsequently, there are several methods to optimize the loss function. Among the most common are:

- **Stochastic Gradient Descent and SGD with Momentum**
  SGD is one of the most common optimization algorithms, used to minimize the loss function. It minimizes the loss function using the gradient computed with respect to the model weights. The gradient indicates the direction of maximum increase of the loss function, and the optimizer updates the weights in the opposite direction, thus reducing the loss value.[76] SGD updates the weights based on the current gradients, which can cause oscillations and high variance during optimization. For this reason, SGD with momentum is introduced, which helps accelerate convergence in the correct direction and dampens oscillations.[75,76]
  The method adds a new variable, called velocity, which combines the current gradients with previous ones, weighted by a decay factor γ (generally between 0 and 1, with a typical value of γ = 0.9).
  The update equations are:

$$v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta) \quad (27)$$

$$\theta = \theta - v_t \quad (28)$$

where, as in the previous paragraphs, η is the learning rate, $\nabla_\theta J(\theta)$ is the gradient of the loss function, and $v_t$represents the velocity. [75,76]

- **Root Mean Square Propagation (RMSProp)**
  RMSProp is an optimizer that dynamically adapts the learning rate for each parameter, useful in the presence of noisy or highly variable gradients. The weight update is performed by dividing the gradient by the square root of the exponential moving average of the squared gradients.[75]

$$E[g^2]_t = \beta E[g^2]_{t-1} + (1-\beta) g_t^2 \tag{29}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t \tag{30}$$

  where β is the decay rate, typically set to 0.9. The square root of $E[g^2]_t$ corresponds to a decaying root mean square of the gradients.[75,76]

- **Adaptive Moment Estimation (Adam)**
  Adam is an optimizer that computes adaptive learning rates for each parameter by leveraging both the first moment (the mean) and the second moment (the variance) of past gradients, at iteration t:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \tag{31}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \tag{32}$$

  where $m_t$is the mean and $v_t$the variance of the gradients. The parameters $\beta_1$and $\beta_2$are exponential decay rates, typically set to 0.9 and 0.999, respectively. [75,76]
  Since the moment estimates are initialized at zero, a bias toward zero is introduced, especially in the first updates. To correct this, bias-corrected estimates are computed:

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \hat{v}_t = \frac{v_t}{1-\beta_2^t} \tag{33}$$

  where $\hat{m}_t$and $\hat{v}_t$are the bias-corrected moment estimates. $\beta_1^t$and β_2^trepresent $\beta_1$and $\beta_2$raised to the power t, respectively. [75,76]
  Finally, the weights are updated according to:

$$\theta_{t+1} = \theta_t - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{34}$$

where $\eta = 0.001$and $\epsilon = 10^{-8}$. [75,76]

In addition to optimizers, effective training of a neural network also depends on the management of the learning rate, the way weights are initialized, the normalization of activations, and regularization techniques that prevent overfitting.
The learning rate(lr) can be set to a fixed value or can adapt automatically during training through a scheduler. Among the most common scheduler strategies are:

- **Step Decay (StepLR)**: reduces the lr by a fixed factor at regular intervals. It is useful when the training process is known, requiring both the decay factor and the interval width to be defined. An example of the lr trend is shown in Figure 44 A.[77]
- **Exponential Decay (ExponentialLR)**: reduces the lr by multiplying it by a decay factor at each epoch (e.g., a factor of 0.95).This schedule allows for large updates at the beginning and progressively smaller updates toward the end of the training.. An example of the lr trend is shown in Figure 44 B.[77]
- **Cosine Annealing (CosineAnnealingLR)**: the lr follows a cosine function, allowing higher values during the initial phases and much lower values in the final ones. An example of the lr trend is shown in Figure 44 C. [77]
- **Adaptive Plateau Reduction (ReduceLROnPlateau)**: reduces the lr when validation metrics show no improvement. It does not require prior knowledge of the training trend but automatically decreases the learning rate when the validation performance remains stable for a certain number of epochs. An example of the learning rate trend is shown in Figure 44 D. [77]
- **Cyclical Learning Rates (CyclicalLR)**: modifies the lr by making it oscillate between a maximum and a minimum value, following a triangular pattern and helping the model escape local minima. An example of the lr trend is shown in Figure 44 E. [77]

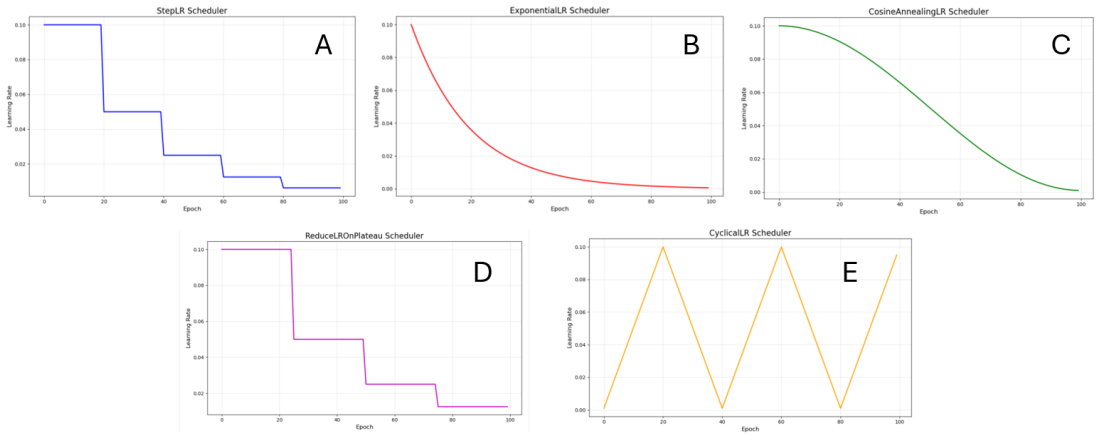

Figure 44: Example of learning rate schedules used in neural network training. A) Step Decay, B) Exponential Decay, C) Cosine Annealing, D) ReduceLROnPlateau, and E) Cyclical Learning Rate. Each schedule illustrates how the lr changes over training epochs.[77]

Initially, the model may converge towards local minima. Choosing inappropriate weights can lead to exploding gradients when the weight values are too large, or to vanishing gradients when the weight values are too small. [78]
Several weight initialization techniques exist:

1. **Random Initialization**: weights are randomly drawn from a normal or uniform distribution with a small standard deviation. [78]
2. **Xavier Initialization**: initializes the weights to maintain a constant variance of signals across layers and of gradients during backpropagation. It works well with symmetric activation functions, such as tanh. [78]
3. **He et al. Initialization**: designed for ReLU activation functions, it accounts for the fact that ReLU zeroes out all negative values. This technique proposes an initial value, called the He initialization. It is similar to Xavier initialization but not identical, as it uses different scaling factors to compensate for the nonlinearity of ReLU. [78]
4. **LeCun Initialization**: also known as efficient backpropagation, it takes into account the input and output dimensions of each layer. It can also be used with activation functions that are not symmetric around zero, such as tanh, generating scaled weights that allow stable signal propagation. [78]

A technique that improves model training is Batch Normalization. During training, the distributions of the inputs to each layer change due to the updates of the parameters of the previous layers. This phenomenon is called internal covariate shift, and to avoid it, a careful choice of all the initial parameters of the network is required. The aim is to reduce internal covariate shift as much as possible. Batch normalization mitigates this effect by normalizing the inputs of each layer over mini-batches of data, so that they have zero mean and unit standard deviation. This technique decreases sensitivity to weight initialization, allows the use of higher learning rates, and accelerates model convergence.[79]
Finally, to improve the model's generalization capability and prevent overfitting, there are regularization techniques such as dropout. Dropout consists of a temporary removal of units in a neural network, along with their incoming and outgoing connections. Each neuron has a probability p of being turned off, and the decision on which neurons are deactivated is random, based on this probability. p is a hyperparameter defined during training. In this way, the network does not rely on individual nodes and becomes more robust.[80]
Another implicit form of regularization is early stopping, a very common method to prevent overfitting. Early stopping consists of selecting as the best model the one corresponding to an epoch earlier than the final one. To work, it monitors the model's performance on the validation set, and the model is selected when the performance on the validation set deteriorates or the loss function starts increasing instead of decreasing.[81]
The operation of early stopping is shown in the Figure 45.

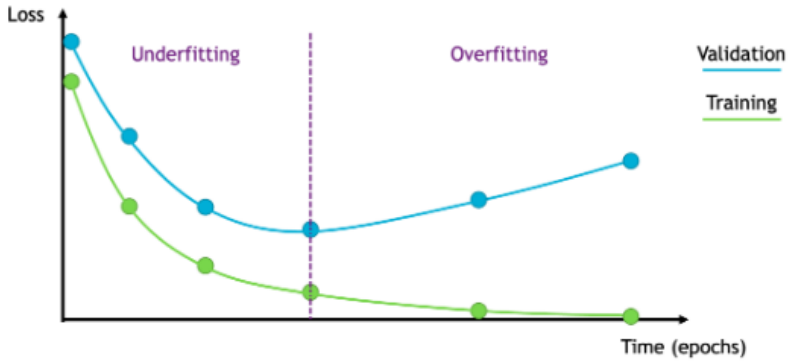

Figure 45: Early stopping selects the model at the point where the validation loss starts increasing, balancing underfitting and overfitting.[81]

All these techniques contribute to making the training of neural networks more stable and improving the model's ability to generalize to unseen data.Inizio moduloFine modulo

3.2.4 Convolutional Neural Network

After presenting fully connected neural networks and the main optimization and generalization strategies, it is natural to introduce Convolutional Neural Networks (CNNs), a deep learning architecture primarily used in the field of image processing. While ANNs struggle to handle the high computational complexity required for processing large-scale data, CNNs are specifically designed to address this challenge.[82] CNNs can be classified according to the dimensionality of their input data: one-dimensional CNNs (1D-CNNs) are based on temporal or sequential signals, represented as numerical values over time, two-dimensional CNNs (2D-CNNs) process inputs such as images, and three-dimensional CNNs (3D-CNNs) are applied to volumetric data, such as 3D images.

Images are composed of basic elements called pixels in two-dimensional images and voxels in three-dimensional images. [83] Each pixel or voxel has an associated numerical value that indicates the brightness intensity of that area or volume of the image. Thus, the main difference between 1D, 2D, and 3D CNNs lies in the spatial arrangement of the input data. In all cases, however, the network receives numerical values as input.

The most common example of CNN application involves 2D images, but the same principles can be applied to 1D signals or 3D volumetric data. As previously described, CNNs include input layers, which receive the initial information, hidden layers, which correspond to Convolutional Layers, Pooling Layers, and output layers which perform the classification and typically include Fully Connected Layers and Non-Linearity Layers, all illustrated in the Figure 46. [84]

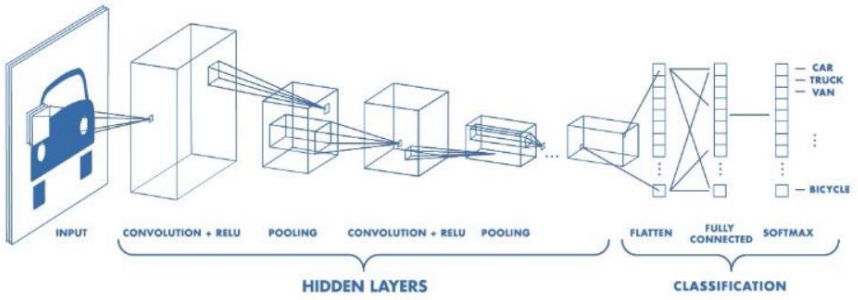

Figure 46: Architecture of a CNN.[84]

The following section provides a detailed description of the layers mentioned above.

- **Convolutional Layer**: This is the fundamental layer of CNNs, from which these networks take their name. Its main function is to extract key features, precisely by using the convolution operation. [84]
  The convolution operation is mathematically defined as the integral of the product of two functions f and g, after one of them is flipped and shifted relative to the other:[81]

$$h(t) = \int_{-\infty}^{\infty} f(t-\tau)g(\tau)d\tau \quad (27)$$

This operation can also be denoted using an asterisk (*):

$$h(t) = (f * g)(t) \quad (28)$$

For discrete and finite signals, such as digital images, the convolution is expressed as a discrete sum: given a discrete signal f[k] and a kernel g[k], with $k \in Z$:

$$h[k] = \sum_{n} f[k-n]g[n] \quad (29)$$

This operation can be extended to multi-dimensional signals. For example, for an image $I[i,j]$ and a kernel $K[i,j]$:

$$H[i,j] = \sum_{m}\sum_{n} I[i-m, j-n]K[m,n] \quad (30)$$

Typically, the first signal is the input of interest, the image, while the second signal is relatively small and implements a specific operation. The second signal is called the kernel.[81]
During convolution, the kernel slides over the image, and the operation produces an activation map $H[i,j]$, containing the most relevant visual features of the image. [84]

The operation is illustrated more clearly in the following Figure 47.[84]

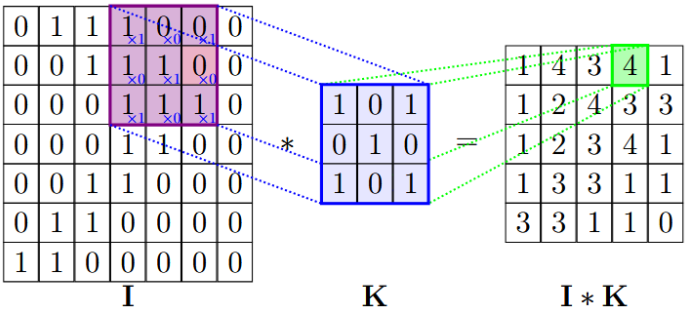

Figure 47: Convolutional layer: A visualization of a discrete convolution operation in 2D.[81]

A convolutional layer is also defined by the following characteristics:

1. Kernel size: indicates the size of the filter (kernel). [84]Variations in the kernel size lead to changes in the size of the resulting feature map.[81]
2. Padding: is the thickness of the zero border added around the input feature map.[81] The padding operation is illustrated in Figure 48.

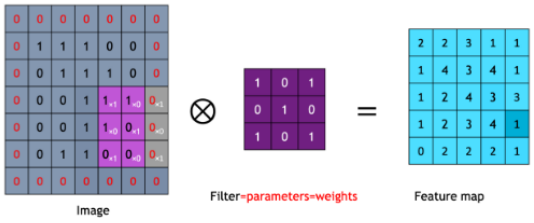

Figure 48: A visualization of the padding operation.[81]

3. Stride: refers to the number of steps the kernel takes before computing the convolution and generating the output pixels. [84]Instead of applying the convolution at every position, using stride = 1, larger step values can be used, resulting in an output feature map with fewer elements.[81] The padding operation is illustrated in Figure 49.

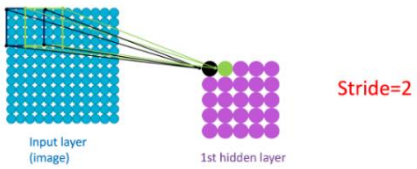

Figure 49: A visualization of the stride operation.[81]

Each convolutional block contains multiple kernels, and to build a network, several convolutional blocks are stacked.

The parameters within the kernels are trained by the CNN, which requires far fewer parameters compared to MLPs. In fact, the kernel is shared across all input positions, significantly reducing the memory needed to store the network parameters. This principle is called parameter sharing or weight sharing.

The process of computing the kernels for each layer and the weight sharing is illustrated in the following Figure 50, where the feature maps are stacked along

an additional dimension. This creates a three-dimensional array when the input is a 2D image, and a four-dimensional array when the input is a 3D image.[81]

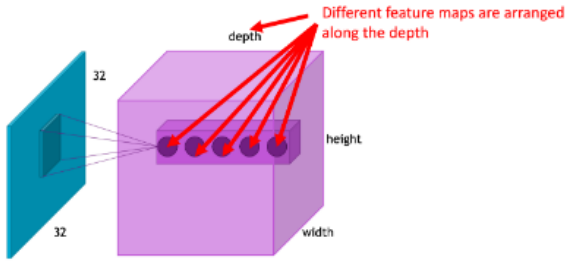

Figure 50: The different feature maps for a given layer are arranged along another dimension.[81]

- **Pooling Layer**: is a layer designed to reduce the number of parameters stored in memory by the network and to decrease the computational load by performing downsampling of the feature maps[84].

  The main operations that can be performed are:

  1) Max Pooling: selects the maximum value within each region, ignoring all other values.[85]
  2) Average Pooling: calculates the average value within each region.[85]

     The choice of the pooling region size is very important:

     - Choosing a large region results in a significant reduction of the feature map, higher computational efficiency, but also a greater loss of information.
     - Choosing a small region preserves more details but increases computational complexity.

     Moreover, these operations result in a loss of spatial information: it is no longer possible to know the original positions of the values within the feature map.[85]
  3) Global Average Pooling: applicable only at the end of the convolutional network, it replaces the fully connected layer. The input consists of multiple feature maps, and for each of them the average of all values is computed, producing a vector in which each element represents the average of one feature map. The resulting vector is then passed to the final activation function.[86]

- **Fully Connected**, or convolutional output layer, is a fully connected layer that receives as input the output of the last pooling layer, flattened by the flatten operation into a vector. Similar to a feedforward neural network, this layer takes the flattened vector as input and produces the final output of the network.[84] Flatten layer and Fully connected layer are shown in the Figure 51.

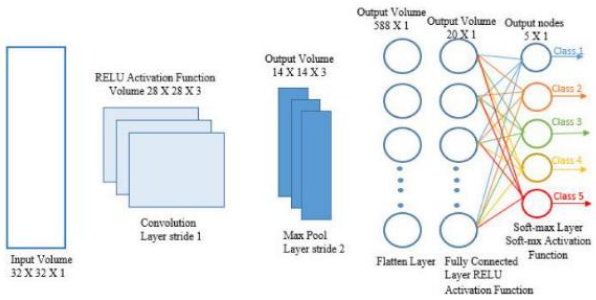

Figure 51: An image showing the main layers of a CNN, including the Flatten layer and the Fully Connected layer.[84]

- **Nonlinearity Layer** (Activation Function) is applied in various layers of the network to introduce non-linearity, through the use of the activation functions described in the previous chapter. This layer also corresponds to the final layer, allowing the network to determine the final output.[84]

3.2.5 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are designed to process sequential data.[72] Such data can include handwritten data, genomes, text, numerical series, or images, if treated as sequences.[87] They consist of an input layer, a hidden layer, and an output layer, and are characterized by recurrent connections, which allow information to circulate within the network. [72] This enables the network to take into account not only the current input but also previous inputs.
At each time step t, the RNN receives an input vector x_t and updates its hidden state h_t using the following equation:

$$h_t = \sigma_h(W_{xh}x_t + W_{hh}h_{t-1} + b_h), \quad (31)$$

Where W_{xh} is the weight matrix between the input and hidden layer, W_{hh} is the weight matrix for the recurrent connection, b_h is the bias vector, and σ_h is the activation function, typically tanh or ReLU. [72]
The output at each time step t is given by:

$$y_t = \sigma_y(W_{hy}h_t + b_y), \quad (32)$$

where W_{hy} is the weight matrix between the hidden layer and the output layer, b_y is the bias vector, and σ_y is the activation function of the output layer.

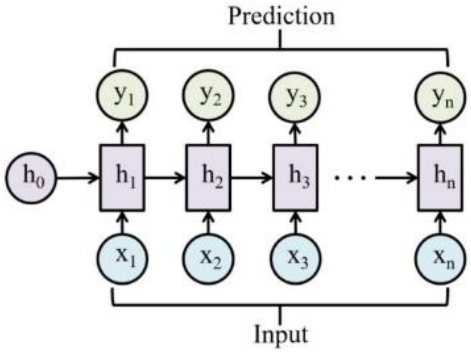

Figure 52: illustration of RNN architecture.

RNNs compute the hidden state by integrating the current input with the previous hidden state. In this way, they are well-suited to maintain dynamic temporal information. The choice of activation function is crucial for the behavior of the network, introducing nonlinearity that allows the model to learn complex patterns in the data.[72]

The update of the hidden state in RNNs can be seen as a function

$$h_t = f(x_t, h_{t-1}), \tag{33}$$

which captures the dependencies between the input sequence and the recurrent connections.

The training of RNNs is performed using the Back Propagation Through Time (BPTT) algorithm, which is used to compute the gradients of the loss function with respect to the weights. As the gradients are propagated backward in time, they can vanish or explode, making it difficult for the network to learn long-term dependencies or causing instability during training.[72,87]

Mathematically, the hidden state at time step ttt can be expanded as follows:

$$h_t = \sigma_h(W_{xh}x_t + W_{hh}\sigma_h(W_{xh}x_{t-1} + W_{hh}h_{t-1} + b_h) + b_h), \tag{34}$$

When computing the gradients, terms involving the product of many Jacobian matrices are encountered:

$$\frac{\partial h_t}{\partial h_{t-n}} = \prod_{k=t-n}^{t-1} J_k \tag{35}$$

where J_k is the Jacobian matrix of the hidden state at time step k. [72]If the eigenvalues of J_k are less than 1, the product of these matrices tends to zero as n increases, leading to the vanishing gradient problem and limiting the contribution of states from much earlier time steps to the current step. Conversely, if the eigenvalues of J_k are greater than 1, the gradients can grow exponentially, causing the exploding gradient problem. This can make the model parameters unstable, excessively modify the weights, cause

the model to converge too quickly to a suboptimal local minimum, or completely fail the training process due to excessively large updates.[72,87]

RNNs can also be implemented in a bidirectional configuration, known as Bidirectional Recurrent Neural Networks (BiRNNs). This architecture processes the input sequence in both forward and backward directions, enabling the network to access information from both past and future contexts. Such a capability improves performance in tasks where understanding the entire sequence is essential, including named entity recognition, machine translation, and speech recognition. The architecture of BiRNNs therefore consists of two hidden states: one for the forward pass and one for the backward pass.[72]

To overcome the vanishing gradient problem, Long Short-Term Memory (LSTM) networks were introduced by Hochreiter and Schmidhuber.[72] LSTMs employ gating mechanisms that regulate the flow of information, enabling the network to preserve long-term dependencies through an internal cell state c_t, which acts as long-term memory. [88] Each LSTM cell manages two types of memory:

- Short-term memory, represented by the hidden state h_t.
- Long-term memory, represented by the cell state c_t. The cell state can be thought of as a "conveyor belt" that carries information across time steps. [72,88]

The main idea of an LSTM cell is to regulate the update of long-term memory so that information and gradients can flow unchanged across iterations.[88]
The Figure 53 shows a representation of an LSTM cell.

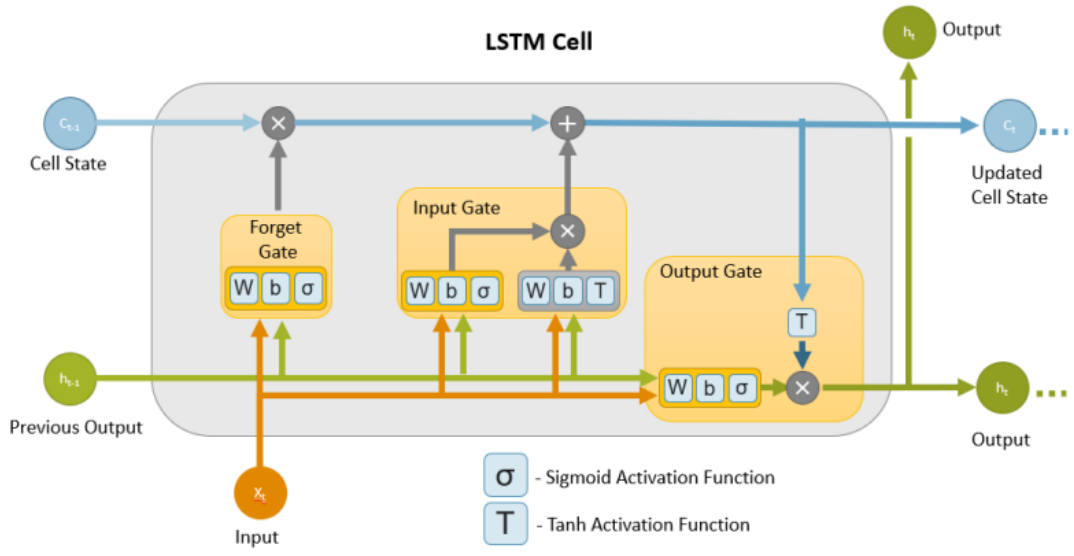

Figure 53: Architecture of a LSTM Cell.[88]

The content of the cell state c_t is updated through three main gates:

- Forget gate: Determines which information from the previous cell state c_{t-1} should be forgotten and which should be retained.[72] The previous state c_{t-1} is updated via

element-wise multiplication with the output of the forget gate f_t. The forget gate uses a sigmoid activation function, ensuring that the output vector has values ranging continuously from 0 (forget) to 1 (keep).[88]

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{36}$$

$$c_f = c_{t-1} \odot f_t \tag{37}$$

Here, $\odot$ denotes element-wise multiplication.[72]

- Input gate: Controls the storage of new information, i.e., how much of the new x_t is written into the cell state c_t.[72] This gate is composed of two functional units:
  - The first uses a $tanh$ activation function, which produces values between -1 and 1 and decides the variation of the cell state $\tilde{c}_t$.[88]

$$\tilde{c}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \tag{38}$$

  - The second uses a sigmoid activation function, which determines the intensity of the variation m_t.[88]

$$m_t = \sigma(W_m \cdot [h_{t-1}, x_t] + b_m) \tag{39}$$

After multiplying the results of these two units, the input gate adds the result to the cell state, updating it.[88]

$$c_t = c_f + \tilde{c}_t \odot m_t \tag{40}$$

- Output gate: Combines long-term and short-term information to provide a more accurate prediction of the next hidden state/output h_t. The output gate uses a trained weight matrix W_o and a bias b_o to extract relevant information from the current input and the previous hidden state. This information is combined with the updated cell state c_t to generate the next output h_t. This output is then recursively used in the next iteration. If the network has multiple layers, the output is also used as input to the next layer; otherwise, it represents the final prediction.[88]

$$o_t = \sigma(W_o[h_{t-1,}x_t] + b_o) \tag{41}$$

$$h_t = o_t \odot \tanh(c_t) \tag{42}$$

Similarly to RNNs, LSTMs can also be organized into bidirectional networks, known as Bidirectional Long Short Term Memory (BiLSTM), which are able to process the sequence in both forward and backward directions, allowing the network to capture context from both the past and the future more comprehensively. In BiLSTMs, for each

time step, two separate hidden states are maintained: one for the forward pass $\overrightarrow{h_t}$ and one for the backward pass $\overleftarrow{h_t}$. In this way, the network processes the input sequence in both directions, maintaining separate hidden states for each direction.[72] The Figure 54 shows a representation of a BiLSTM network.

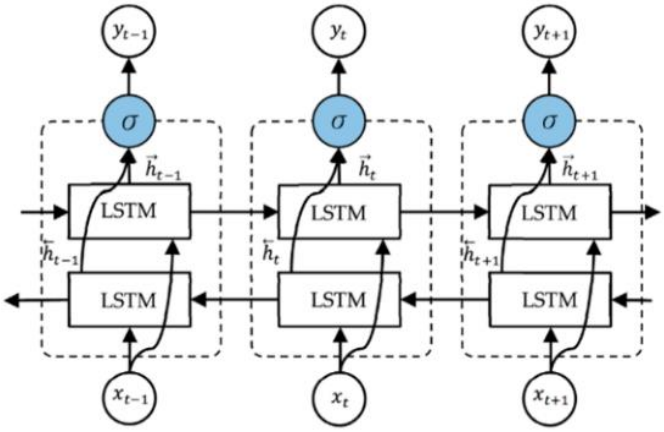

Figure 54: Architecture of BiLSTM network.[72]

3.2.6 Transformers for Time-Series

The Transformer architecture was introduced by Vaswani et al., a group of researchers from Google Brain, in 2017. It is the first architecture based entirely on attention mechanisms, completely eliminating both recurrence and convolutions. The first Transformer was applied to automatic text translation from English to German and English to French, achieving performance superior to all existing models at the time. Subsequent applications include reading comprehension, sentence embedding learning, and text generation.[89]

The self-attention mechanism at the core of Transformers allows the model to relate different positions within a single sequence, thereby producing a global representation of the entire sequence.[89]

The Transformer architecture follows an encoder-decoder structure: the encoder maps a sequence of input symbol representations $(x_1, \dots, x_n)$ into a sequence of continuous representations $z = (z_1, \dots, z_n)$, while the decoder generates an output sequence $(y_1, \dots, y_n)$ from z, producing one symbol at a time. The model is autoregressive, meaning that at each step it uses the previously generated symbols as input when producing the next symbol. [89] An illustration of the architecture is shown in Figure 55.

Delving into the architecture, the encoder consists of six identical layers, each composed of two sub-layers: the first is a multi-head self-attention mechanism, and the second is a fully connected feed-forward network. Each sub-layer includes a residual connection, which adds the input to the output of the sub-layer, followed by layer normalization. [89]The final output of a sub-layer is thus:

$$LayerNorm(x + sublayer(x)) \quad (43)$$

The decoder also consists of six identical layers, each composed of three sub-layers: masked multi-head self-attention over the decoder outputs, multi-head attention over the encoder's output, and fully connected feed-forward network.

The masked multi-head attention prevents positions from attending to future tokens. Moreover, the output embeddings are shifted by one position, ensuring that each token prediction depends only on previous positions. As in the encoder, residual connections and layer normalization are applied around each sub-layer.[89]

Below is a brief explanation of classical attention and multi-head attention:

The basic attention function, called scaled dot-product attention, is defined as:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^\top}{\sqrt{d_k}})V \quad (44)$$

where $Q, K \in \mathbb{R}^{d_k}$ and $V \in \mathbb{R}^{d_v}$. An alternative, less common version used in Transformers is additive attention.[89]

In multi-head attention, the queries, keys, and values are linearly projected h times, producing different representations. Attention is applied in parallel to each representation, generating outputs of dimension d_v. The outputs from all heads are then concatenated and linearly projected using a learned weight matrix W_O, which combines the outputs into the final representation:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)\, W_O \quad (45)$$

Where:

$$\text{head}_i = \text{Attention}(QW_Q^i, KW_K^i, VW_V^i) \quad (46)$$

and W_Q^i, W_K^i, W_V^i are learned projection matrices. In the original Transformer, 8 heads were used.

Thanks to the attention mechanism, Transformers allow for greater parallelization and shorter training times compared to traditional recurrent or convolutional networks.[89]

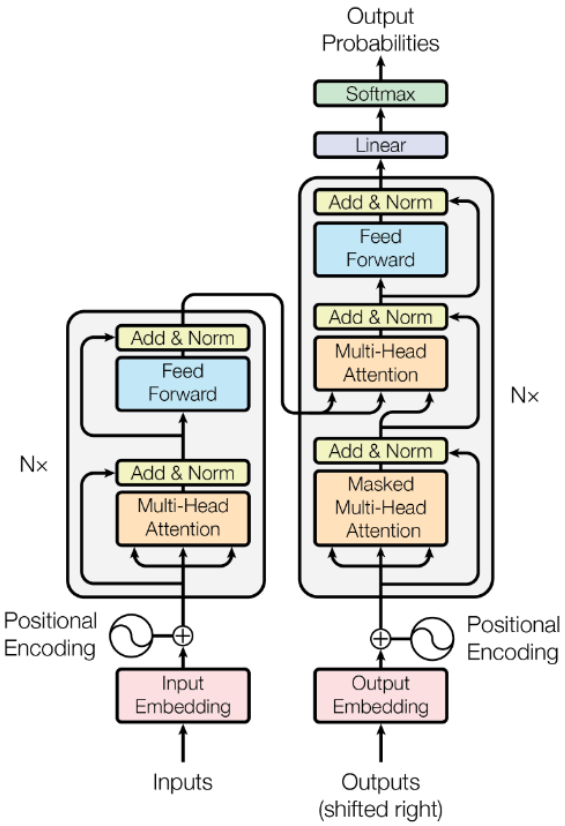

Figure 55: Architecture of the Transformer.[89]

3.2.7 Self-Supervised Learning for Signals: Wav2vec and Wav2vec 2.0

The wav2vec network was introduced in 2019 with the aim of learning representations of human speech directly from raw audio, without the need for manual transcriptions, thereby reducing the need for large annotated datasets for automatic speech recognition (ASR). It is a self-supervised model that leverages unsupervised pre-training on speech. The learned representations are then used to improve the training of the acoustic model.[90]

Starting from the raw audio input, the model is trained to predict future audio samples based on the given context, in order to capture the structures of speech. The ultimate goal is to distinguish a true future audio sample from false or distractor samples.[90]

The model consists of two main components:

1. Encoder network: embeds the audio signal into a latent space. Given a raw audio sample $x_i \in X$, the encoder transforms it into a latent vector $f: X \mapsto Z$.
   Encoder network is composed of five convolutional layers with kernel sizes (10, 8, 4, 4, 4) and strides (5, 4, 2, 2, 2). The output is a low-frequency representation $z_i \in Z$, encoding approximately 30 ms of 16 kHz audio, with representations z_i produced every 10 ms.[90]

2. Context network: combines multiple temporal steps from the encoder to obtain contextualized representations.
   The network $g: Z \mapsto C$ takes latent representations $z_i \dots z_{i-v}$ as input and combines them into a contextualized tensor $c_i = g(z_i \dots z_{i-v})$.
   It has nine convolutional layers with kernel size 3 and stride 1, with a total receptive field of about 210 ms.[90]

Both the encoder and context network layers use causal convolutions with 512 channels, a group normalization layer, and a ReLU nonlinearity. Normalization is applied across both feature and temporal dimensions to ensure stability and robustness across different datasets.[90] An illustration of the architecture is shown in Figure 56.

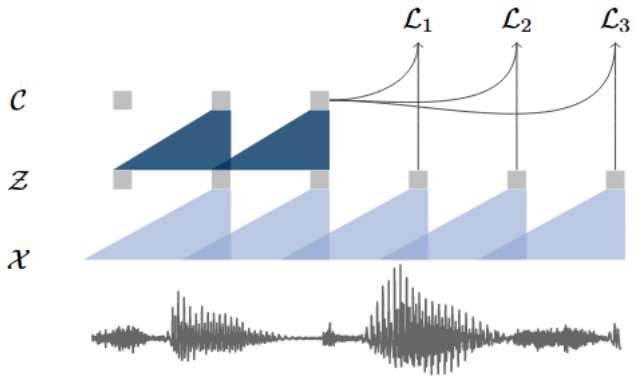

Figure 56: Wav2vec pre-training: audio X is processed by two stacked CNNs, and the model is trained to predict future time steps, learning speech representations.[90]

After training, the representations c_i produced by the context network are used as input to the acoustic model, replacing traditional log-mel filterbank features. These "intelligent" representations allow the acoustic model to convert sounds into written words, acting like an automatic translator from audio to text.[90]

Wav2vec 2.0, is an evolution of this model and introduces several key innovations, including the use of Transformers and discrete quantization of audio features.
The model processes raw audio through a multi-layer convolutional encoder, which transforms the signal into latent representations capturing local speech information. These representations are then fed into a Transformer, whose task is to build contextualized representations that integrate information across the entire temporal sequence.[91]
Wav2vec 2.0 consists of three main components, which work in sequence to convert raw audio into contextualized embeddings suitable for pre-training and, subsequently, for fine-tuning on supervised speech recognition tasks.

- A multi-layer convolutional encoder $f: X \mapsto Z$ that takes raw audio $x_i \in X$ as input and produces latent speech representations $z_1, \dots, z_T$ for T time steps. The encoder is composed of several blocks, each consisting of a temporal convolution followed by layer normalization and a Gaussian Error Linear Unit(GELU) activation. The input waveform is normalized to zero mean and

unit variance. The total stride of the encoder determines the number of time steps T that are fed into the Transformer.[91]

- Quantization $Z \mapsto Q$: the continuous features produced by the encoder are converted into discrete representations q_t via a product quantization module. This step defines a finite set of discrete speech units, which serve as targets for pre-training.[91]
- A Transformer $g: Z \mapsto C$: it processes the latent representations $z_1, \ldots, z_T$ from the encoder to produce contextual representations $c_1, \ldots, c_T$, integrating temporal information across the entire audio sequence.[91]

During pre-training, a certain proportion of time steps in the encoder's latent feature space is masked. The training objective is to predict, for each masked step, the correct quantized representation q_t of the audio from a set of distractors.
Finally, the model is fine-tuned on labeled data for supervised speech recognition tasks.[91]
An illustration of the architecture is shown in Figure 57.

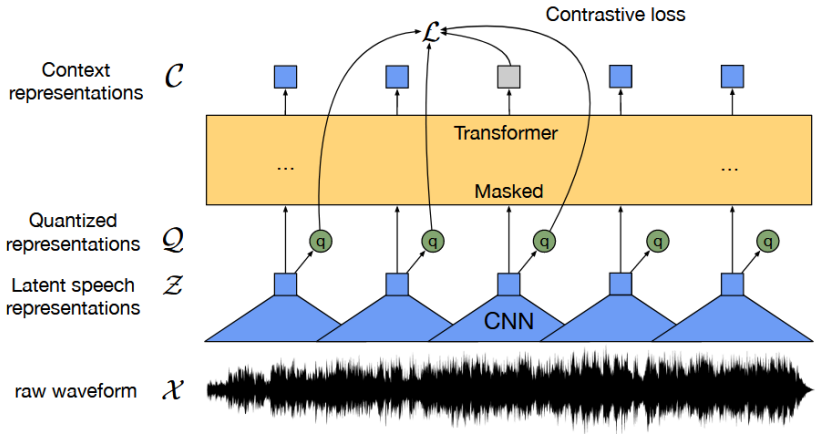

Figure 57: Illustration of Wav2vec2.0. [91]

3.3 3D Imaging Analysis

3.3.1 Deep Learning Approaches for 3D Imaging Analysis

In recent years, the integration of deep learning and neuroimaging has provided new opportunities for the early diagnosis and monitoring of neurodegenerative diseases. Three-dimensional convolutional neural networks (3D-CNNs) have shown great potential for the automatic analysis of volumetric brain data. This chapter describes several studies that have applied 3D-CNN models for the classification and recognition

of major neurodegenerative disorders, including Alzheimer's disease, dementia, and Parkinson's disease, paving the way for potential applications in the study of PPA.

1. **Automated classification of Alzheimer's disease and mild cognitive impairment using a single MRI and deep neural networks.** [92]

   This study developed a deep learning algorithm based on T1-weighted magnetic resonance images, capable of distinguishing healthy controls (HC) from patients with Alzheimer's disease (AD) and classifying subgroups of -) into stable (s-MCI) and progressive/converting (c-MCI).
   The main dataset used was the ADNI database, comprising 1,409 subjects: 294 patients with probable AD, 763 patients with MCI, and 352 healthy controls. All baseline 3D T1 scans were included. Additionally, an independent external validation dataset was created, consisting of 229 subjects recruited at the Neurology Department of the Scientific Institute and Vita-Salute San Raffaele University in Milan (124 with probable AD, 50 with MCI, and 55 healthy controls).
   The images underwent preprocessing, including normalization to a standard space, followed by data augmentation to increase data variability and prevent overfitting. The data augmentation operations applied included deformation, flipping, scaling, cropping, and rotation. The dataset was divided into training, validation, and test sets, with 90% of the images allocated to training and validation and 10% to testing. A 10-fold cross-validation approach was used to optimize the network.
   The proposed network is a 3D CNN composed of 12 repeated convolutional blocks: the first 2 blocks use 50 kernels of size 5×5×5 with alternating strides of 1 and 2, while the subsequent 10 blocks use 100 to 1,600 kernels of size 3×3×3 with alternating strides of 1 and 2. Max pooling layers were replaced with convolutions with stride 2. The network includes ReLU activation layers, one fully connected layer, and an output layer.
   Performance was evaluated across several comparisons, including AD vs HC, c-MCI vs HC, s-MCI vs HC, AD vs c-MCI, AD vs s-MCI, and c-MCI vs s-MCI. The data were split into three stages: training, validation, and testing. Images were divided into 90% for training and validation and 10% for testing. Data augmentation was applied to selected training and validation images to generate additional images and prevent overfitting. The applied data augmentation operations included deformation, flipping, scaling, cropping, and rotation. A 10-fold cross-validation approach was applied to identify the optimal network.
   The results demonstrated high accuracy: the AD vs HC comparison reached 99.2%, while the c-MCI vs s-MCI comparison achieved the lowest value of 74.9%. Other comparisons fell within this range.
   In conclusion, the proposed 3D CNN demonstrated promising capabilities for the automatic and early detection of AD, suggesting a potential role for structural MRI in clinical practice.

2. Detection of Dementia Through 3D Convolutional Neural Networks Based on Amyloid PET. [93]

This study proposes a three-dimensional convolutional neural network (3D-CNN) for the automatic detection of dementia based on amyloid Positron Emission Tomography(PET) images, recognized as a key biomarker of neurodegeneration associated with dementia.

The dataset used is OASIS-3, which includes MRI and PET images of 1,098 participants, comprising 605 healthy subjects and 493 individuals with varying degrees of cognitive decline, aged between 42 and 95 years. The dataset contains a total of 1,607 PET scans, of which approximately 1,352 were used in this study.

The images underwent preprocessing as follows: PET scans, acquired as time series, were averaged along the temporal dimension to obtain a single 3D mean image per subject. This approach allows PET scans to be treated as regular MRI/CT volumes and helps reduce noise. Subsequently, Otsu thresholding was applied to segment the brain region, identifying a bounding box around the brain. Images were then resized to 128×128 voxels, corresponding to the minimum size in the dataset, and only the central 50 slices of each image were retained, as these were considered the most informative.

The developed model is a 17-layer 3D-CNN, consisting of four convolutional layers with 64, 128, and 256 filters, all using 3×3×3 kernels. Each convolutional layer is followed by a max pooling layer with stride 2 and a batch normalization layer. The final part of the model includes a dense layer with dropout and an output neuron with a sigmoid activation function for binary classification (dementia vs. control).

The model was trained using the cross-entropy loss function, optimized with the Adam algorithm. The mean accuracy achieved was approximately 83%, indicating that 3D convolutional networks applied to amyloid PET scans represent a promising approach to support early diagnosis and recognition of Alzheimer's disease. The architecture of the proposed model is shown in Figure 58.

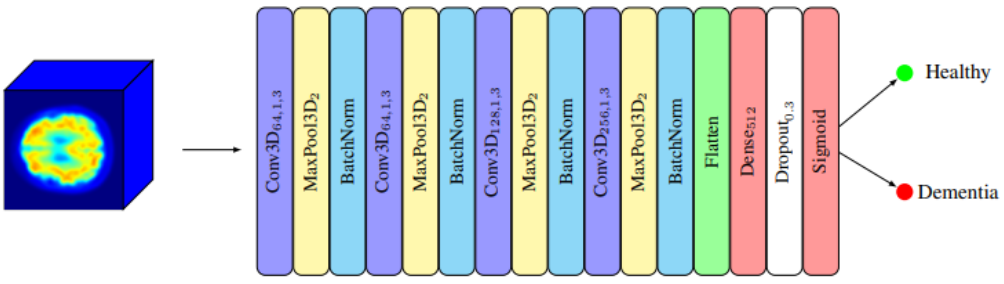

Figure 58: Architecture of the proposed 3D-CNN model. [93]

3. Multi-Center 3D CNN for Parkinson's disease diagnosis and prognosis using clinical and T1-weighted MRI data.[94]

The aim of this study was to develop a three-dimensional convolutional neural network (3D-CNN) based on T1-weighted MRI images to distinguish patients with Parkinson's disease (PD) from healthy controls and, subsequently, to predict PD progression.

The dataset consisted of three independent multicenter cohorts with T1 MRI images and clinical follow-up assessments (for at least 2 years) for PD patients and healthy controls:

- Cohort A: 86 patients with mild PD, 62 with moderate-to-severe PD, and 60 controls (1.5T MRI).
- Cohort B: 14 patients with mild PD and 14 controls from the Parkinson's Progression Markers Initiative database.
- Cohort C: 38 de novo patients with mild PD and 38 controls (3T MRI).

PD patients were classified into two progression clusters (stable vs. worsening) using k-means clustering on clinical variables, such as baseline and follow-up UPDRS-III scores.

A 3D CNN was built and tested on patients and controls for binary classifications:

- Controls vs. moderate-to-severe PD
- Controls vs. mild PD
- Two PD progression clusters (slow vs. fast)

The 3D CNN takes T1-weighted images as input, without significant preprocessing, and outputs the subject groups. The architecture consists of 8 repeated convolutional blocks (3×3×3 kernels with alternating strides of 1 and 2), a ReLU, a fully connected layer, and a softmax output layer. Max pooling layers were replaced with standard convolutional layers with stride 2.

A transfer learning approach was applied, where the weights of the CNN trained to distinguish controls from moderate-to-severe PD patients were used for initialization and subsequently updated via gradient descent. To improve generalization, data augmentation was applied to the images, including deformation, flipping, scaling, cropping, and rotation, each with a 50% probability.

Performance:

- Controls vs. moderate-to-severe PD: 74.04% accuracy; with data augmentation: 71.2%
- Controls vs. mild PD: without transfer learning: 50.28%; with transfer learning: 64.27%; with data augmentation: 58.30%
- Progression clusters (slow vs. fast): using clinical data only: 51.16%; adding demographic and clinical information (disease duration, baseline UPDRS-III, age, sex): 72%; with transfer learning: 65.62%

This study demonstrated that 3D CNNs could become a promising tool to study Parkinson's disease and its progression. The architecture of the proposed 3D-CNN model is shown in Figure 59.

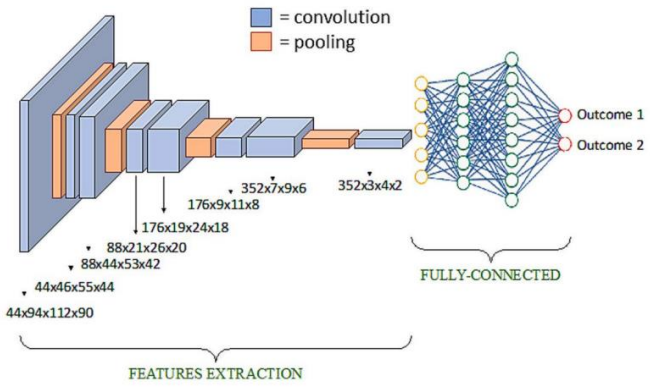

Figure 59: Architecture of the proposed 3D-CNN model.[94]

3.4 Speech Analysis

3.4.1 Deep Learning Approaches for Speech Analysis

Speech disorders have been extensively studied both in the medical field and in scientific literature. However, the use of human speech, obtained from voice recordings, is still rarely employed as a source of information for deep learning networks. One of the main challenges in this field is the lack of complete datasets, due to the difficulty in performing speech tests on patients with neurological problems. The physical and cognitive problems of these individuals can complicate voice recordings and introduce high variability, making it difficult to ensure consistent data with high quality.[95]

Most of the existing approaches are based on ML techniques, which exploit manual features extraction. Only a few approaches use deep learning as a classification method, despite it representing the most effective solution, capable of capturing non-linear relationships and complex temporal dependencies that are difficult to model with traditional approaches.[96]

This section shows a list of existing studies in scientific literature, which exploit the information of human language and apply deep learning techniques for classification purposes. These studies also describe the dataset used, the pre-processing performed, the approach applied and the performance achieved.

1. **Deep Learning and Artificial Intelligence Applied to Model Speech and Language in Parkinson's Disease.**[97]

   This study proposes the automatic assessment of PD patients using different approaches based on speech, language and their combination.

The database used consists of 165 audio recordings of native Spanish-Colombian patients, of which 80 suffer from PD. The audios contain a description of a typical day, lasting about 90 seconds. The recordings are sampled at a frequency of 44.1 kHz and underwent two pre-processing steps: Global System for Mobile Communications (originally Groupe Spécial Mobile, GSM) compression at full speed and downsampling to 8 kHz. Audio transcriptions were obtained through an Amazon transcription service for HC and PD patients.

Methods proposed in this article initially treat speech and transcription as independent. The following approaches are used for speech:

- 1D-CNN: a model using one-dimensional convolutional layers followed by Long Short Term Memory layers, with the aim of extracting temporal information in the time domain.
- 2D-CNN working on time-frequency representations: the audio signals are divided into 500 ms blocks with a phase shift of 250 ms, then transformed using the Short-Time Fourier Transform (STFT) and converted into Mel scale spectrograms. These are then processed by a 2D CNN organized according to the ResNet architecture, which allows the model to capture relevant patterns in both time and frequency domains through residual blocks and pooling, obtaining compact embedding to provide the final classification layer.
- Wav2vec 2.0: a model based on pre-trained transformers trained on large amounts of unlabelled data, which directly encodes speech audio using a multilayer convolutional network followed by a transformer that constructs contextualized representations. These representations are then temporally pooled to obtain a single vector representation of the signal, which is finally used for classification through a fully connected layer.

Three word-embedding models are used for the language. These models associate a corresponding vector to the single word:

- Word to Vector (W2V): a model that generates word embeddings independent of context.
- Bidirectional Encoder Representations from Transformer (BERT): a model that generates word embeddings that are context dependent.
- Bidirectional Encoder Representations from Transformers for Spanish (BETO): Spanish version of BERT.

Embeddings are processed using two approaches:

- Using a statistical representation that uses mean, standard deviation, asymmetry and kurtosis to then train a SVM model.
- Using a CNN that receives in input the embedding matrix and applies one-dimensional convolutions with kernels of different sizes to capture bi-gram, tri-gram and four-gram relationships between words. This is followed by a max grand pooling and a fully connected layer that performs the final classification.

Among these models, the best for speech is Wav2vec 2.0, achieving an accuracy of 88.5%, while for language it is BETO + CNN, with an accuracy of 77.9%.

These two models were then merged through multimodal fusion techniques, to improve performance by integrating information from both audios and transcriptions. The fusion approaches considered were: early fusion, which combines the features of both modalities at the input level, joint fusion, which combines intermediate representations of models during training and late fusion, which integrates the output of the two modalities at the classification level.
The best approach consists of joint fusion with an accuracy of 77.2%. However, the fusion approach had lower accuracy compared to the speech-only approach. This could be attributed both to the way the embeddings were applied and to the fact that PD affects more the production of the language, than the language itself.
The Figure 60 shows a graphical pipeline of the approach applied.

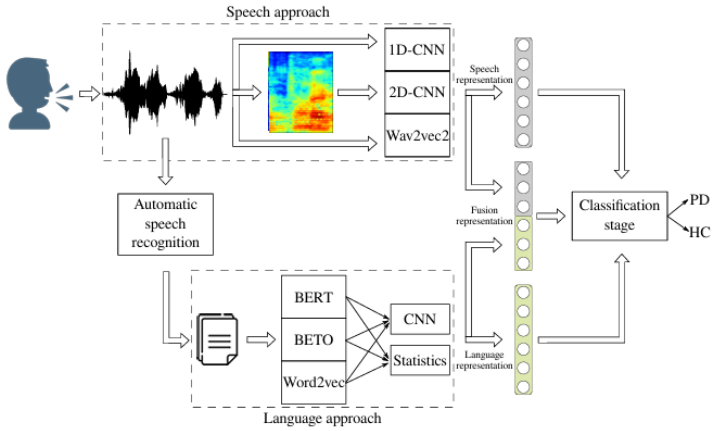

Figure 60: Graphical pipeline of the article. [97]

2. Evaluating raw waveforms with deep learning frameworks for speech emotion recognition. [98]

This article focuses on the recognition of human emotions, comparing traditional ML techniques with DL approaches.
Six different databases are used in the study:

- Berlin Database of Emotional Speech (EMO-DB), which includes emotions such as neutrality, happiness, anger, disgust, sadness, boredom, and fear/anxiety.
- Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS), with emotions such as surprise, anger, happiness, boredom, sadness, fear, disgust, and neutrality.
- Toronto Emotional Speech Database (TESS), which includes anger, disgust, neutrality, fear, happiness, sadness, boredom, and surprise.
- Crowd-sourced Emotional Multimodal Actors (CREMA), with emotions of sadness, happiness, disgust, neutrality, anger, and fear.
- Surrey Audio Visual Expressed Emotion (SAVEE), which includes surprise, anger, happiness, disgust, neutrality, fear, and sadness.
- TESS+RAVDESS, a combination of the two datasets TESS and RAVDESS.

In the first part of the article, conventional feature extraction techniques such as the Mel-scaled spectrogram and the Mel-Frequency Cepstral Coefficients (MFCC) are applied, followed by ML algorithms including SVM, k-NN, decision trees, Naive Bayes, majority voting, and stacking. These results are then compared with the performance achieved by DL techniques.

Before applying DL techniques, each dataset is normalized to have a mean of 0 and a variance of 1. Audios with a duration of more than 6 seconds are truncated, while those with a shorter duration are completed by zero-padding to standardize their length.

Three different deep learning architectures are used in the article:

- 1D-CNN, that automatically extracts audio features using local filters.
- LSTM, which models temporal dependencies within audio sequences thanks to long-term memory mechanisms.
- CNN-LSTM, a hybrid model which combines the automatic feature extraction of CNNs with the ability of LSTMs to capture long-term temporal dependencies.

The network outputs are eight, as the maximum number of emotions that exist in the dataset.

The results reported in the article show that the mediated accuracies on all databases are higher for DL models compared to traditional ML approaches. Among all the evaluated models, the one achieving the best performance is 1D-CNN, with results for each dataset present in Table 4. These results show that long-term dependencies features seem to play a minor role in speech emotion recognition.

3. Multimodal deep learning for dementia classification using text and audio. [99]

This article is based on the use of both speech and transcriptions of patients suffering from dementia to classify their condition, because dementia also causes problems related to verbal deficits. The dataset used is DementiaBank's Pitt Cookie Theft, which consists of audio recordings of dementia patients describing a picture of a cookie theft.

To process the audio recordings, Wav2vec was used as feature extractor, while pre-trained Word2vec from the "word2vec-google-news-300" dictionary in Gensim25 were used to process the transcription data and map each to its associated embedding. Words without embedding were marked as Out-of-Vocabulary (OOV) and represented by zero vectors. For each word, the timestamp (start and end times) was preserved and the start time of each sentence is modified to 0.

Four types of datasets have been created at the text preprocessing level:

- Original dataset.
- Original dataset with the removal of sentences shorter than two words.
- Augmented datasets using synonyms replacement.
- Augmented datasets using synonyms replacement with the removal of sentences shorter than two words.

Models were initially created by processing audio and text separately:

- The first model is created to classify audio only and is based on the Wav2vec architecture. It generates audio embeddings that are processed through a pooling and dropout layer, followed by a fully connected dense network, able to perform binary classification.
- The second model is created for text classification, it uses Word2vec to generate embedding vectors, which are then processed by an LSTM network, followed by a dense layer for classification.
- The third approach combines the audio model with the use of timestamps, which are placed in input to an LSTM network that integrates this information with audio embedding, before switching to the dense net.
- In the fourth approach, timestamps are linked directly to textual embeddings and the entire sequence is processed by an LSTM and then by a dense network.
- A fifth approach uses the audio model merged with the text model.
- The sixth one combines the text model with the timestamp information and the audio model.

These models are applied using the four different types of datasets as textual input. The best result was obtained with the fourth approach, which integrates timestamps with textual embedding using an LSTM network, achieving an accuracy of 84.78%.

4. Deep Learning of Speech Data for Early Detection of Alzheimer's Disease in the Elderly.[100]

This article focuses on the most common form of dementia, Alzheimer's, a pathology characterized by progressive deterioration of language skills.

The dataset consists of recordings from 80 patients: 40 healthy controls and 40 dementia patients. All patients underwent the Mini-Mental State Examination for Dementia Screening (MMSE-DS), but only the voice responses to 12 selected questions, which require immediate and spontaneous verbal answers, were retained for analysis.

Preprocessing consisted of extracting patients' voice from the video files using ffmpeg, with a sampling rate of 44.1 kHz, stereo and MP4 format to reduce information loss. As a second step, the duration of the response is fixed to 3 seconds; if the response is shorter (which always happens, average duration: 1–2 seconds), moments of silence are padded. If the answer does not arrive, a time up to 30 seconds is given and if the answer is still not given, everything is reduced to 3 seconds of silence.

The 3-second audio clips are then converted into spectrograms using the MFCC, which generate 12 spectrograms, one for each response, which were then merged into a single MFCC image. A data augmentation is applied to the images in order to increase the dataset ten times, using brightness adjustment and horizontal shifts.

Five CNN models are trained to classify the audio data:

- Densenet121, a network that uses dense connections between layers, improves the propagation of the gradient and promotes the reuse of features, useful for capturing MFCC patterns.
- Inception v3, a CNN featuring Inception blocks that extract characteristics at different scales, identifying local and global patterns in voice images.
- VGG19, a deep and simple architecture that captures stable patterns in spectrograms.
- Xception, a network that uses deep separable convolutions, making the analysis of complex and non-linear MFCC patterns efficient.
- Resnet50, a network that leverages residual connections, allowing you to train deep networks and capture relevant patterns even in the presence of variability between patients.

These five models are evaluated through:

- 5-fold cross-validation, which shows that Densenet121 and VGG19 achieve the highest performance with an accuracy of 90%, followed by Inception v3 with an accuracy of 87.5%.
- Hold-out validation, which shows that Resnet50 achieves the highest performance with an accuracy of 93.75%, followed by Inception v3 and Densenet121 with an accuracy of 87.5%.

The study demonstrates that even brief voice responses can provide sufficient information for classification of Alzheimer's disease.

The Figure 61 shows a graphical pipeline of the applied approach.

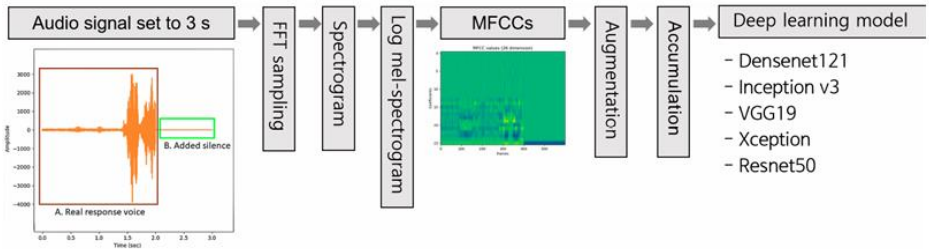

Figure 61: Graphical pipeline of the article. [100]

5. Speech task based automatic classification of ALS and Parkinson's Disease and their severity using log Mel spectrograms.

This study is based on the use of a deep learning approach applied to patients' voice recordings in order to diagnose and assess the severity of their disease.

Specifically, the dataset consists of 60 patients affected by amyotrophic lateral sclerosis (ALS), 60 patients with PD, and 60 HC. The recruited patients come from the National Institute of Mental Health and Neurosciences in Bangalore, India, and performed four different types of vocal tasks: spontaneous speech or monologue (SPON), image description (IMAG), production of specific phonemes (PHONE), and sentence repetition or dictation (DIDK). Patient audio recordings were initially sampled at 44.1 kHz and then downsampled to 16 kHz. Each audio recording was

subsequently segmented into varying lengths, specifically 0.5, 0.8, 1, 1.2, 1.5, 2, and 3 seconds, with an overlap of 0.1 s.

Regarding the network architectures, three types of networks were trained:

1. A network capable of diagnosing between ALS, PD, and HC.
2. A network able to assign the severity level of ALS, trained to classify five ALS severity classes.
3. A network to assign the severity level of PD, trained for three classes.

For all three approaches, log-mel spectrograms (SPEC) and MFCC coefficients were used and compared to determine the better-performing approach. Both representations were provided as input to a 2D-CNN, as illustrated in Figure 62.

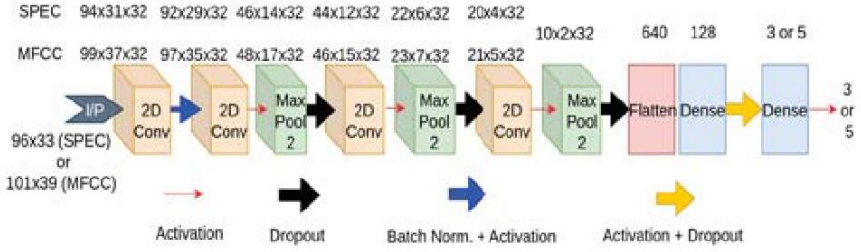

Figure 62: Illustration of the 2D CNN architecture used for classification.[101]

A five-fold cross-validation approach was used to train the models. The study demonstrates that SPEC outperforms MFCC for all tasks.

The following Table 4 shows a summary of the techniques applied in the various articles, making it easier to compare them.

Ref	Dataset	Exam	Preprocessing	Approach	Results
[97]	165 participants: - 80PD - 85HC	90s audio recording of regular day description	Audio: -GSM compression - 8 kHz downsampling	Speech: - 1D-CNN - 2D-CNN - Wav2Vec2.0 Text: -Word2Vec+CNN -Word2Vec+statistics -BERT+CNN -BERT+statistics -BETO+CNN -BETO+statistics Fusion: Wav2Vec2.0+BETO+CNN - Early fusion - Joint fusion - Late fusion	Acc. Speech: 88.5% Acc. Text: 77.9% Acc. Fusion: 77.2%

[98]	6 Datasets: - EMO-DB - RAVDESS - TESS - CREMA - SAVEE - TESS+RAVDESS	6s audio recordings of emotional speech	- Audio normalization to mean 0 and variance 1 - Truncation of audio longer than 6s - Zero padding for audio shorter than 6s	Speech: three models are tested - CNN - LSTM - CNN+LSTM	6 Datasets' accuracy for the best model CNN: - Acc.: 90.34 - Acc.: 90.42 - Acc.: 99.46 - Acc.: 69.72 - Acc.: 85.76 - Acc.: 95.86
[99]	Pitt Cookie Theft dataset from DementiaBank	Audio description of Cookie Theft Image	Audio and Text: Data divided into individual sentences. Each word is associated with start and end timestamps. Text: Different preprocessing conditions - Original data - Data with short sentences removed - Text-based augmentation of the original data - Text-based augmentation of the data with short sentences removed Timestamps: The starting timestamp of each sentence is normalized to zero.	Speech: - Wav2vec+Dense layer Text: - Word2vec+ LSTM + Dense layer Time: - Timetsamp+ LSTM + Dense layer Early fusion: Text+Time Speech+Time Speech+Text Speech+Text+Time	Text based augmentation + Text + Time Acc.: 84.51 +- 0.3
[100]	80 participants: - 40AD - 40HC	12 audio responses during the MMSE-DS exam	- Audio extraction from video with characteristics of 44.1 kHz, stereo, MP4 - Audio length standardized to 3s - Zero padding if answers are shorter than 3s - If no answer within 30 s, replaced with 3 s of silence - Conversion of 3s clips into MFCC spectrograms	- Densenet121 - Inceptionv3 - VGG19 - Xception - Resnet50	Best performances: 5 fold cross validation: - Acc. Densenet121 90% - Acc.Inceptionv3 87.5% - Acc.VGG19 90% Hold out validation - Acc. Resnet50 93.7% - Acc.Inceptionv3 87.5%

			- Union of 12 spectrograms to create a single image - Data augmentation using brightness adjustment and horizontal shift.		- Acc. Densenet121 87.5%
[101]	80 participants: -60ASL -60PD - 60HC	4 voice tasks: -SPON -IMAG -PHONE -DIDK	- Resampling of audio from 44.1kHz to 16 kHz. - Audio length standardized to 3s - Audio segmentation with window lengths w = {0.5, 0.8, 1, 1.2, 1.5, 2, 3} s and 0.1 s overlap. - Conversion into MFCC and SPEC spectrograms	- SPEC+ CNN2D - MFCC+ CNN2D	Best performances: 5 fold cross validation: Best performance obtained with SPEC+ CNN2D

Table 4: Table summarizing the articles analysed previously.

3.4.2 Current Approaches for the Automatic Classification of PPA

This section presents various studies in the literature that, starting from audio recordings of patients with PPA, are able to detect the disease and classify it into its variants.

All the existing approaches leverage ML algorithms that require manual feature extraction followed by the training of various traditional classifiers.

In the 2013 study by Fraser et al., Automatic Speech Recognition in the Diagnosis of Primary Progressive Aphasia [102], the dataset used consisted of audio recordings of PPA patients obtained through picture description tasks. The dataset was fully transcribed, combining automatic transcriptions with manual transcriptions. Acoustic and textual features were extracted from both the audio and the transcriptions, and several types of classifiers were trained to output the two most well-known variants at that time: semantic dementia and progressive non-fluent aphasia. The classifiers trained included SVM, Naive Bayes, and Random Forest.

In the 2021 study by Themistocleous et al., Automatic Subtyping of Individuals with Primary Progressive Aphasia[103], a dataset of 44 patients with PPA was used. The

dataset consisted of audio recordings of the Cookie Theft picture description from the Boston Diagnostic Aphasia Examination. The audio recordings were transcribed using Themis, a widely used tool for clinical transcriptions. The approach then involved manual extraction of acoustic and linguistic features, which were fed into a classifier. For classification, a MLP was used, a feed-forward neural network that takes the concatenated acoustic and linguistic features as input and outputs the predicted class. The model consisted of eight hidden neurons and used the ReLU activation function. The trained model achieved an accuracy of 80%, correctly identifying lvPPA patients in 95% of cases, svPPA patients in 65% of cases, and nfvPPA patients in 90% of cases. At present, all existing approaches rely on ML algorithms with manual feature extraction. The lack of studies using deep learning techniques in this context allowed the development of this thesis.

3.5 Multimodal fusion

Multimodal fusion refers to the integration of complementary features from multiple data streams, with the goal of combining different types of information to achieve more accurate predictions. A method is defined as multimodal when it leverages multiple types of data to solve a problem.[104]

The first multimodal approach dates back to 1989, in an audio-visual speech task applied to speech signals degraded by noise. In this study, a neural network combined visual and acoustic information to improve speech perception.[104]

Several types of multimodal fusion exist:

- **Early Fusion**: Data from different modalities are combined at the input stage of each branch, before being processed by the model, creating a richer and more informative input. In other words, data are fused at the input level and passed through a single encoder to extract complex and integrated features.[104,105] This strategy is also known as data-level fusion.[105] The process is illustrated in Figure 63.

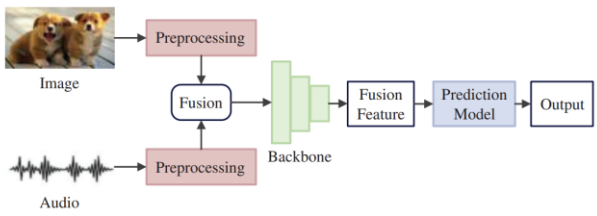

Figure 63: An example of early fusion. [104]

- **Deep Fusion**: Fusion occurs during the feature extraction phase, combining information in the feature space. This strategy is also known as feature-level fusion. This approach allows the use of features extracted at different levels of abstraction in deep networks, enhancing model performance in some

applications, although it may sometimes result in lower performance compared to early fusion.[104,105] The process is illustrated in Figure 64.

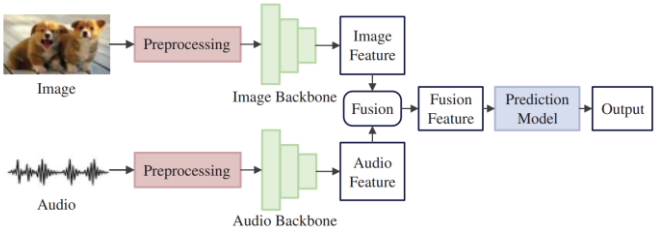

Figure 64: An example of deep fusion. [104]

- **Late Fusion**: This is a decision-level fusion performed at the prediction stage. Each modality is processed by a separate branch, and the final decision is obtained by combining the outputs of the different models. This approach, also called model-level fusion, aims to optimize the final response and improve overall accuracy.[104,105] The process is illustrated in the Figure 65.

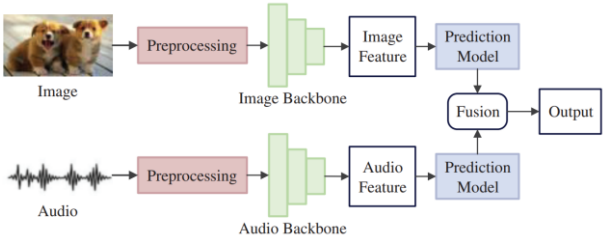

Figure 65: An example of late fusion. [104]

- **Hybrid Fusion**: This combines information at different levels, for example, data or features from one branch with decisions from another branch, establishing a cascade relationship. In this way, hybrid fusion combines the advantages of early, deep, and late fusion strategies, but it significantly increases model complexity and training difficulty.[104] The process is illustrated in Figure 66.

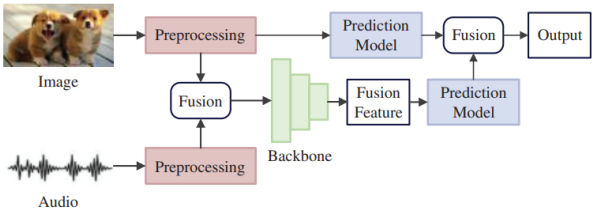

Figure 66: An example of hybrid fusion.[104]

3.5.1 Deep Learning Approaches for Multimodal Fusion

In the literature, various multimodal approaches have been proposed that leverage the integration of information from multiple modalities to improve classification performance. Several examples have already been discussed in the previous sections,

such as in the paper Deep Learning and Artificial Intelligence Applied to Model Speech and Language in Parkinson's Disease[97], in the paper Multimodal Deep Learning for Dementia Classification Using Text and Audio[99], and in the article Multi-Center 3D CNN for Parkinson's Disease Diagnosis and Prognosis Using Clinical and T1-Weighted MRI Data [94], in the phase of disease progression analysis.
In this section, further multimodal approaches will be presented.

1. **PD-Net: Parkinson's Disease Detection Through Fusion of Two Spectral Features Using Attention-Based Hybrid Deep Neural Network.[106]**

   This study focuses on Parkinson's disease, a progressive neurodegenerative disorder that causes, among other symptoms, vocal impairments, often appearing as some of the earliest signs of the disease. The article investigates the use of voice for early diagnosis, specifically leveraging a combination of two spectral features: the Mel spectrogram and MFCCs. The proposed model employs a hybrid neural network composed of a CNN for spatial feature extraction and an LSTM for temporal feature extraction. Additionally, a multi-head attention mechanism enhances the model's ability to focus on the most relevant details of the vocal signal.
   The initial dataset consisted of 831 audio recordings, including: reading phonetically balanced texts, uttering syllables such as "pa" and "ta," phonation of the vowels ("a," "e", "i", "o", "u"), and reading phonetically balanced words and phrases. The audio recordings were segmented into 3-second windows and processed using cleaning, normalization, and data augmentation techniques, including pitch scaling, random gain modification, and addition of white noise, resulting in a total of 3496 audio segments. For each audio segment, the Mel spectrogram (FFT of 2048 and hop length of 512) and MFCCs (converted into 30 Mel bands) were computed to capture the key characteristics of the signal. Features extracted by the CNN were concatenated at the feature level and passed through a hybrid attention model, followed by the LSTM, to extract temporal features. L2 regularization was applied to reduce overfitting.
   The final classification task differentiates between Parkinson's patients and healthy subjects. The proposed model, named PD-Net, achieved an accuracy of 99%. In conclusion, the study demonstrates that integrating advanced spectral features with deep learning techniques represents a promising approach for the early diagnosis of Parkinson's disease. Figure 67 illustrates the framework of the study.

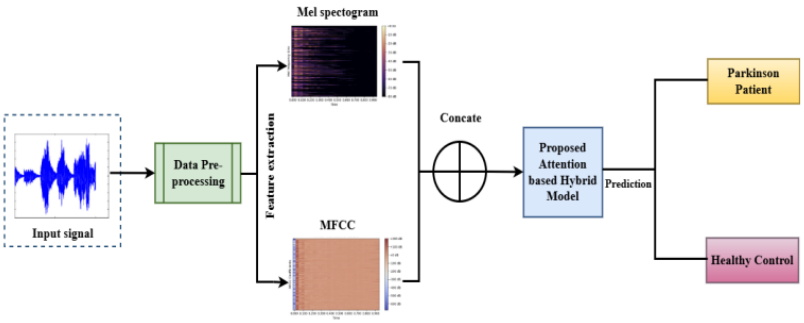

Figure 67: Abstract view of the proposed Parkinson disease detection framework through vocal analysis. [106]

2. Multimodal Fusion of EEG and Audio Spectrogram for Major Depressive Disorder Recognition Using Modified DenseNet121.[107]

This study aims to classify Major Depressive Disorder (MDD) using information derived from both EEG signals and voice recordings. EEG, which measures the brain's electrical activity, has proven useful in diagnosing neurological, cognitive, and psychological disorders. Voice recordings, on the other hand, reflect language alterations typical of depression, such as slowed speech rate, monotone prosody, and frequent pauses, indicative of psychomotor retardation and cognitive slowing. The objective is to develop a multimodal classification model that integrates EEG and audio data to identify depressive tendencies.

The dataset includes:

- EEG data: a total of 53 participants, of whom 24 were diagnosed with MDD and 29 were HC.
- Audio data: a total of 52 participants, of whom 25 had MDD and 27 were HC. Participants were asked to read a text, describe an image, and answer open-ended questions.

The data were preprocessed as follows:

- EEG signals were transformed using the STFT.
- Audio signals were transformed into Mel-spectrograms.

For feature extraction, pre-trained DenseNet121 networks were applied, with the convolutional layer weights frozen to extract invariant features. To enable multimodal fusion, the final layer of each network was modified: a fully connected layer was removed and replaced with a layer that combines the features from the EEG and audio branches, followed by the final classification layer. This approach constitutes a transfer learning strategy, in which only the final layer was trained to distinguish between HC and MDD.

The multimodal classification model achieved excellent performance, with Accuracy of 97.53%, Precision of 98.20%, F1-score of 97.76% and Recall of 97.32%

These results demonstrate that integrating EEG and vocal data significantly enhances classification performance compared to unimodal models, highlighting the effectiveness of multimodal fusion in detecting MDD.

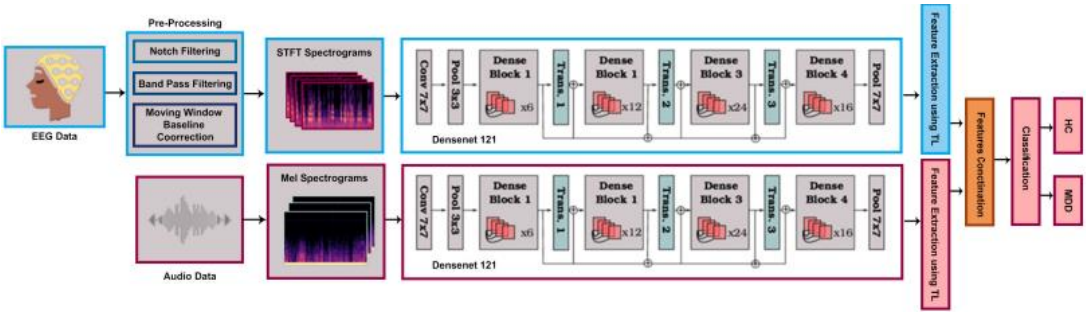

Figure 68: Workflow of the proposed multimodal classification approach. [107]

3. **Early and Late Fusion for Multimodal Aggression Prediction in Dementia Patients: A Comparative Analysis.**[108]

In this article, the strategies of Early Fusion and Late Fusion were compared for the classification of aggression in dementia patients.

The model is based on a multimodal dataset:

- Visual data: collected from the dataset "3D Human Pose in The Wild Using IMUs and a Moving Camera". The images showed situations in which an argument between two individuals escalated into aggression or remained a casual conversation. In each frame, the 3D coordinates of facial, body, and hand keypoints were detected, resulting in a total of 1629 features.
- Audio data: obtained from the "Kaggle: Audio-based Violence Detection Dataset", containing audio clips of aggressive and non-aggressive verbal episodes. The audio files were filtered, and the following features were extracted: MFCC (26 features), pitch (mean and standard deviation, 2 features), spectral features (centroid, bandwidth, flatness, contrast, 4 features), zero-crossing rate (mean, 1 feature) and RMS energy (mean, 1 feature), for a total of 39 features.

A Random Forest classifier was then trained. In the Early Fusion approach, audio and visual features were concatenated and provided as a single input to the classifier, whereas in the Late Fusion approach, two separate models were trained (one for audio and one for visual data), and the probabilities produced by each model were combined using a meta-classifier, also based on Random Forest.

The final output was the classification of aggressive vs. non-aggressive behavior.

The results show that Late Fusion outperforms Early Fusion in terms of accuracy (0.876 vs. 0.828), recall (0.914 vs. 0.818), F1-score (0.867 vs. 0.835), and ROC-AUC (0.970 vs. 0.922). In contrast, Early Fusion demonstrated higher precision (0.852 vs. 0.824), indicating fewer false positives. Moreover, Late Fusion was faster during inference, making it suitable for potential real-time applications.

4. Aim

Primary Progressive Aphasia is a relatively recent neurodegenerative disease. In 2011, the recognition of the three PPA variants and the diagnostic criteria still in use, were published for the first time.
From a diagnostic perspective, patients are evaluated using cognitive and linguistic tests as well as brain imaging, such as magnetic resonance imaging (MRI). This allows clinicians to assess both the patients' linguistic characteristics, identifying errors typical of each variant, and the presence of focal atrophy in the brain regions most affected.
The main objective of this thesis is to demonstrate how a deep learning-based approach can support the diagnosis of PPA, leveraging the same types of information used in clinical practice, namely linguistic data and MRI scans.
The specific objectives of the thesis are as follows:

1. **To develop deep learning models based on patients' audio recordings collected during cognitive tests**. This represents a novel contribution to the literature, as audio-based approaches have not yet been explored for PPA classification.
2. **To develop deep learning models based on MRI scans**. Similar approaches have been applied to Alzheimer's patients, but their extension to PPA has not yet been explored.

The models are designed to perform multiclass classification, distinguishing between the three PPA variants and healthy subjects, and are then applied to all possible binary class comparisons.

3. **To perform multimodal fusion of audio- and MRI-based predictions**, with the aim of demonstrate that integrating these two types of information can improve overall diagnostic performance, effectively replicating the clinical process followed by clinician in an automated manner.

5. Materials and Methods

5.1 Dataset

The study included a total of 94 patients diagnosed with PPA, of whom 38 were diagnosed with nfvPPA, 36 with svPPA, and 20 with lvPPA. Patients were recruited from six reference centers in the Lombardy region and subsequently referred to the IRCCS Ospedale San Raffaele in Milan between 2010 and 2025. A control group was included to ensure the validity of the study, consisting of 91 healthy subjects matched with the PPA patients for age, sex, and educational level. Controls were recruited partly among patients' relatives and partly from individuals who regularly participate in research studies.

The total sample was composed of two population subgroups:

1. The first subgroup included patients who had undergone a neurological examination (including the Mini-Mental State Examination (MMSE), the Beck Depression Inventory (BDI), and the Picnic Scene test), a neuropsychological assessment, and a complete audio recording of the Picnic Scene test from the Western Aphasia Battery (WAB).
   This subgroup included 81 patients, with 34 diagnosed with nfvPPA, 27 with svPPA, and 20 with lvPPA, along with 38 control subjects.
2. The second subgroup included patients and controls who underwent magnetic resonance imaging examinations. This sample comprised 90 patients with 38 diagnosed with nfvPPA, 33 with svPPA, and 19 with lvPPA, and 82 controls.

The two subgroups partially overlapped. Specifically, a subgroup of 77 patients completed both the neuropsychological assessment and the MRI, including 34 with nfvPPA, 24 with svPPA, and 19 with lvPPA , along with 29 control subjects. The two assessments for each participant were conducted within a maximum interval of six months.

5.1.1 Clinical, Neuropsychological, and Language Assessment

Neuropsychological assessments were conducted by experienced neuropsychologists, who were blinded to the MRI results. In all patients, the neuropsychological evaluation investigated:

- Global cognitive functioning using the MMSE

- Frontal/executive functioning using the Frontal Assessment Battery (FAB)
- Verbal memory using the digit span forward and the Rey Auditory Verbal Learning Test (RAVLT)
- Non-verbal memory using the spatial span forward and Rey's Figure delayed recall
- Attention and executive functions using the attentive matrices, digit span backward, Trail Making Test, Colored Progressive Matrices, and Clock Drawing Test
- Visuospatial abilities using the Rey-Osterrieth Complex Figure copy
- Levels of autonomy using the Activities of Daily Living (ADL) and Instrumental Activities of Daily Living (IADL) scales
- Behavior using the Neuropsychiatric Inventory and the Frontal Behavioral Inventory
- Disease severity using the Clinical Dementia Rating (CDR), CDR-Sum of Boxes, and CDR-FTLD (Frontotemporal Lobar Degeneration) scales.

Furthermore, patients underwent a detailed language assessment, which included:

- Syntactic comprehension using the Token Test
- Naming and single-word comprehension using subtests of the CaGi battery
- Object knowledge using the Pyramids and Palm Trees Test
- Repetition, reading, and writing using the Aachen Aphasia Test (AAT)
- Verbal fluency using phonemic and semantic fluency tests.

5.2 Speech Data Acquisition from the Picnic Test

In this thesis, patient voice data were used, obtained through audio recordings of the Picnic Scene test. This is a standardized task employed to assess language and communication disorders and serves as a diagnostic tool for patients with PPA. The Picnic Scene test is part of the Western Aphasia Battery (WAB), a widely used instrument for evaluating both linguistic and non-linguistic abilities in adults with aphasia.[109] It consist of an oral descriptions of an image depicting a picnic scene, known as the Picnic Scene Description Task.

During the task, the patient is presented with an image representing an everyday life situation, specifically an outdoor picnic. Participants are asked to describe the scene spontaneously, providing details and forming complete sentences, typically prompted with: "Look at this picture, tell me what you see, and try to speak in full sentences." The examiner intervenes only when strictly necessary, for instance, to encourage the patient to continue after prolonged pauses.

The language samples consisted of the patients' oral descriptions of the scene and were recorded in audio format using recording devices.

The picnic scene used in the test is shown in Figure 69.

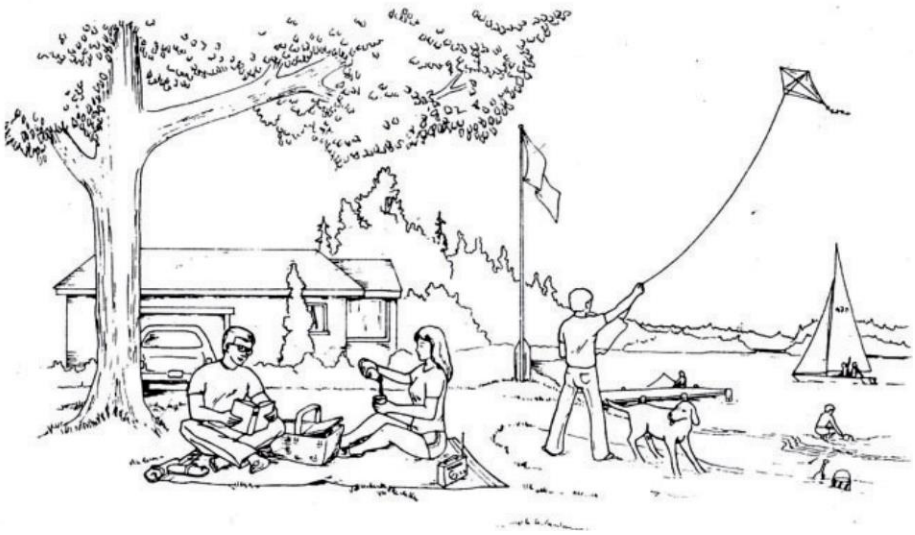

Figure 69: The picnic scene from the Western Aphasia Battery (Kertesz, 1982), which patients were asked to describe. [110]

5.3 MRI Data Acquisition

A total of 161 patients underwent brain MRI on two different 3.0 T Philips Medical Systems scanners (Scanner 1: Intera; Scanner 2: Ingenia CX) at IRCSS San Raffaele Scientific Institute between 2010 and 2025.

- Using Scanner 1, in 40 patients and 30 HC, 3D T1-weighted fast field echo images were acquired (TR = 25 ms, TE = 4.6 ms, flip angle = 30, 220 contiguous axial slices with voxel size = 0.89 × 0.89 × 0.8 mm, matrix size = 256 × 256, FOV = 230 × 182 mm2).
- Using Scanner 2, in 50 patients and 52 HC, 3D high resolution T1-weighted turbo field echo images were acquired (FOV = 256 × 256, pixel size = 1 × 1mm, 204 slices, 1mm thick, matrix = 256 × 256, TR = 7ms, TE = 3.2ms, TI = 1,000ms, FA = 8, ETL = 240, TA = 8.53min).

For acquisitions on both scanners, all slices were positioned to run parallel to a line that joins the most inferoanterior and inferoposterior parts of the corpus callosum.

5.4 Audio-Based Deep Learning Models

This section presents the preprocessing procedures, the AI model architecture, and the post-processing steps adopted to develop deep learning models capable of distinguishing between healthy controls and patients with PPA, as well as recognizing the different variants, using audio recordings collected during the Picnic Scene Task. The code was entirely developed in *Python*, with most of the implementation based on the *PyTorch* library.

5.4.1 Audio Preprocessing Pipeline

The audio recordings were acquired in the hospital using non-standardized devices, so preprocessing was necessary to make all recordings consistent. The following steps were applied during preprocessing:

1. **Standardization of the recording format:** all audio files were converted to the Waveform Audio File Format (WAV). The WAV format is used to store high-quality digital audio, as it preserves data without any compression, maintaining full sound fidelity across different sampling rates and bitrates.[111] The conversion was performed using *FFmpeg*, a leading multimedia framework for handling audio and video files, capable of decoding, encoding, transcoding, muxing, demuxing, streaming, filtering, and playing audio files.[112]
2. **Removal of medical interventions:** during the recordings, doctors occasionally intervened to encourage the patients. These interventions were manually removed using *REAPER*, a comprehensive digital audio production application that provides tools for multitrack recording, editing, processing, mixing, and mastering of audio files.[113] Specifically, during the manual operation, the time window in which the healthcare operator spoke was selected and that segment was removed. In cases where the doctor's voice overlapped with the patient's, both voices were retained to avoid losing useful information from the patient's speech.
3. **Denoising:** Subsequently, a noise removal approach was applied. The recordings contained non-stationary noise, including typical hospital background sounds, ringing cell phones, and noises generated by objects with which patients interacted during the description, such as pens or other surrounding items. The denoising phase was therefore particularly long and complex, aiming to remove as much background noise as possible, so that it would not be provided as input to the artificial intelligence networks used in the subsequent stages. Several approaches were tested, but the two considered valid and implemented in this thesis are:
   - **Denoising Method Using DeepFilterNet2:**
     For noise reduction in audio signals, the neural network *DeepFilterNet2* was used, designed by Schröter et al. This network operates with low latency and reduced computational requirements, making it particularly suitable for real-time applications and for handling non-stationary noise.[114] Several versions of the network are publicly available on GitHub; for this work was used the latest version: DeepFilterNet2.
     The input audio was preprocessed to meet the network requirements: all signals were resampled to 48 kHz using the Python library Librosa, and converted to mono by averaging the stereo channels, an operation automatically performed by Librosa.
     DeepFilterNet2 operates in two main stages. In the first stage, the audio signal is processed to extract and enhance the Equivalent Rectangular

Bandwidths(ERB) bands, modeling human frequency perception and enhancing the vocal envelope of the signal. During this stage, the estimated gains are calculated and applied to the ERB bands, allowing the network to emphasize desired speech components while reducing noise. In the second stage, the complex features of the signal are extracted and enhanced, aimed at strengthening the periodic components of speech. These features are decoded without applying gains, but they contribute to the final improvement of the signal when their output is combined with that of the ERB bands.

The information from both stages is then processed by an encoder, which uses 1D convolutions to capture local temporal information, grouped linear layers (GLinear) to reduce computational complexity, and recurrent units (GRU) to model long-term temporal dependencies.

Subsequently, the signal passes through two separate decoders: one dedicated to the ERB bands and one to the complex features. The ERB decoder uses transposed convolutions (TConv), GLinear, and GRU, and applies the estimated gains to reconstruct the enhanced ERB bands. The complex features decoder, instead, uses only GLinear and GRU to reconstruct the enhanced complex features without applying gains.[114]

The encoder, the ERB decoder, and the Deep Filter(DF) decoder are shown in Figure 70.

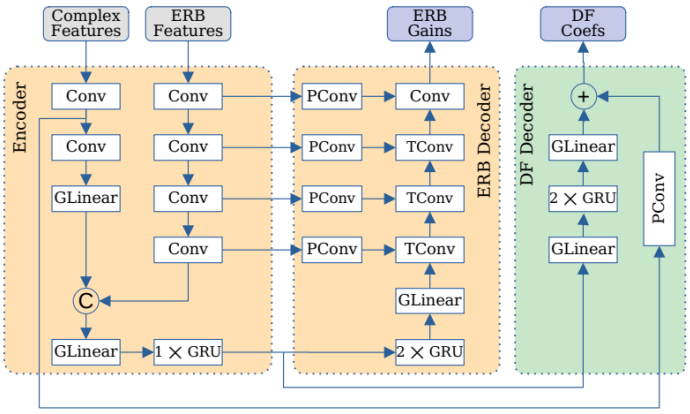

Figure 70: Architecture of the Encoder, ERB Decoder, and DeepFilterNet Decoder of the DeepFilterNet2 network. [114]

The outputs of the two decoders are then combined to produce the final filtered signal. Finally, an inverse STFT (iSTFT) is applied to bring the signal back to the time domain, making the clean audio waveform ready for analysis or further processing.[114]

Figure 71 shows a complete illustration of the two stages of DeepFilterNet2

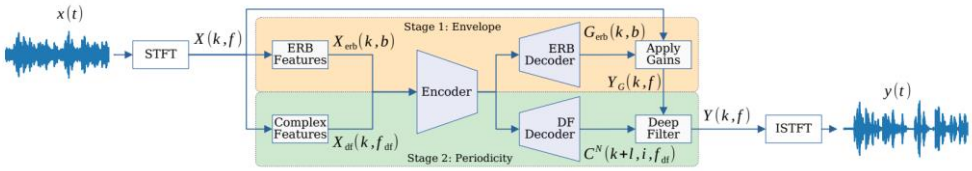

Figure 71: Stages 1 and 2 of the DeepFilterNet2 network: a schematic overview of all steps. [114]

- **Denoising Method Using Demucs:** A second strategy for noise reduction made use of a pretrained model known as Demucs. Demucs is a neural network developed by the Facebook AI Research (FAIR) team in 2019, originally designed for source separation. In this work, the model was employed as a denoising technique by treating noise as an additional source to be separated from the patients' speech signal.

  Unlike DeepFilterNet, Demucs operates directly in the time domain. The network consists of three main components: an encoder, a bi-LSTM, and a decoder, which are connected through a U-Net.

  The encoder consists of six stacked layers. Each layer includes three convolutional operations: the first convolution uses a kernel size of 8, a stride of 4, and a ReLU activation function; the second convolution uses a kernel size of 1, a stride of 1, and a GLU activation; the third convolution uses a kernel size of 3, a stride of 1, and again a ReLU activation.

  The decoder has a symmetric structure with respect to the encoder, and thus also consists of six layers. Each layer includes a convolutional layer with kernel size 3, stride 1, and ReLU activation, followed by a convolutional layer with kernel size 1, stride 1, and GLU activation. .[115]

  The encoder and decoder are connected through a biLSTM layer, which extracts features related to the temporal context of the signal.

  Finally, the model outputs separate waveforms for each source.[115]

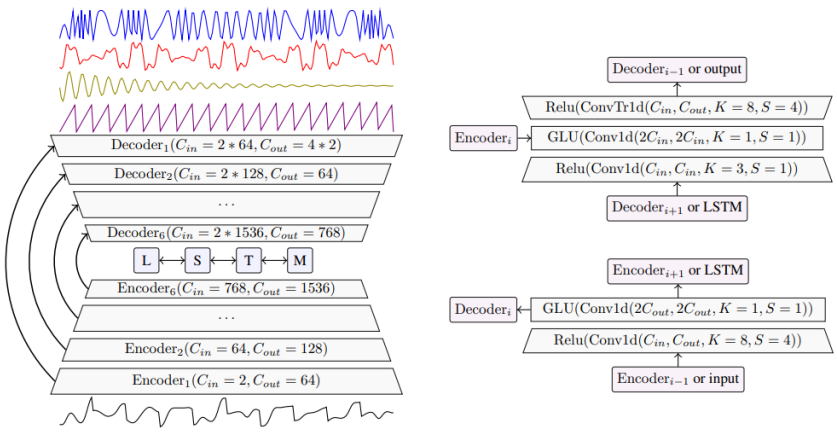

Figure 72: On the left, the Demucs architecture is shown, with a mixture of four waveforms as input and four estimated sources as output. The arrows represent the U-Net skip connections. On the right, a zoomed view of the encoder and decoder layers is presented, with the encoder on top and the decoder at the bottom.

4. **Normalization and Segmentation Strategies:**
   Subsequently, two additional preprocessing steps were applied to the audio signals: normalization and segmentation.
   Focusing on normalization, all audio signals were subjected to z-score normalization. Specifically, for each audio, the mean and standard deviation were computed, and normalization was applied according to the formula:

$$Z = \frac{\text{audio} - \text{mean}}{\text{std}} \quad (43)$$

To avoid numerical instability, a control condition was introduced: if the standard deviation was smaller than 10^{-8}(an arbitrarily chosen value), only the mean was removed, without dividing by the standard deviation:

$$Z = \text{audio} - \text{mean} \quad (44)$$

Next, each signal was amplitude-scaled to the range $[-1,1]$ to ensure further uniformity of the inputs. This approach, widely used in works on human speech and audio signals, has been adopted, for example, in:

- Learn from Real: Reality Defender's Submission to ASVspoof5 Challenge by Zhu, Y. at al.[116]
- Wave-based damage detection in solid structures using a spatially asymmetric encoder–decoder network by Frank Wuttke et al.[117]

The signal was scaled using the following formula:

$$x_{\text{scaled}} = 0.99 \cdot \frac{x}{\max(|\, x \,|)} \quad (45)$$

The factor 0.99 introduces a small safety margin, ensuring that even after subsequent operations performed by the network, no sample exceeds ±1.
This step allows:

1) Make the scale consistent across different samples, facilitating the learning process of the network.
2) Prevent clipping: when the values of a signal exceed the maximum allowed, the exceeding portions are flattened, causing nonlinear distortion. Scaling prevents clipping from occurring in the preprocessing pipeline. However, it should be noted that pre-existing clipping, for instance due to recordings with excessively high peaks, is not corrected by this scaling and will persist in the reduced signal.

As explained in "Nonlinear waveform distortion: Assessment and detection of clipping on speech data and systems" (Hansen et al., 2021),[118] clipping in audio signals is caused by exceeding the maximum representable limits and can generate significant signal distortions.

Examining the segmentation process in more detail, the audio signals were segmented according to a specific procedure. Each recording was divided into fixed-length fragments, with a temporal overlap between consecutive segments. This procedure was applied across the entire duration of each signal.

In cases where the last fragment was shorter than the predefined window length, zero-padding was applied to ensure that all segments from each audio file had the same duration. The configurations considered were as follows:

- 6 s with 1 s overlap
- 6 s with 3 s overlap
- 12 s with 6 s overlap

This approach ensured that all audio inputs provided to the network had a controlled and uniform duration. The order of the fragments was preserved by assigning each segment a progressive identifier during the segmentation process, thus maintaining the original temporal sequence.

The choice of the window length and overlap represented a significant challenge. As discussed in the chapter Deep Learning Approaches for Speech Analysis, most previous studies use windows of 3 seconds or shorter. However, in the case of PPA patients, such short windows would have resulted in excessive fragmentation of speech, potentially splitting even single words. For this reason, slightly longer windows were adopted to preserve more linguistic context within each fragment.

The segmentation with overlap was introduced for three main reasons:

1) Handling input length: neural networks cannot effectively process signals that are excessively long or of variable duration. In the dataset considered, the audio recordings ranged from one to three minutes, which is too long for efficient model processing and contains an excessive amount of information.
2) Preserving speech continuity: introducing an overlap between segments reduces the likelihood that the last word of a sentence will be truncated during segmentation. Thanks to this overlap, it is highly probable that a word partially cut off at the end of one fragment will appear in full in the subsequent one, thus preserving the continuity and coherence of speech.
3) Natural data augmentation: the use of overlap increases the number of fragments available for each patient, implicitly enlarging the amount of data available for network training and improving model generalization.

Overall, this segmentation strategy allowed both to control the length of the inputs and to effectively enrich the dataset.

Finally, two different strategies combining normalization and segmentation were implemented:

a) Normalization followed by segmentation: In this case, the entire audio recording was first normalized and then segmented. The normalization, therefore, depends on the overall structure of the full signal.
b) Segmentation followed by normalization: In this approach, each audio recording was first divided into fragments, and each fragment was then

normalized individually. In this way, normalization depends solely on the structure of each single segment.

5. **Log Mel Spectrograms, Mel Spectrograms and MFCCs Generation**
   Subsequently, the audio fragments were transformed into spectrogram-based representations, including Mel spectrograms, log-Mel spectrograms, and MFCCs, approaches commonly used in several studies described in the previous chapter. The three approaches are described below:

   - Mel spectrogram: as a first step, the spectrogram of the audio fragments is computed, i.e., they are transformed into a time–frequency representation of the signal obtained by applying the Fast Fourier Transform (FFT).
     The frequencies obtained from the spectrograms are then converted into the Mel scale, introduced in 1937 by Stevens, Volkmann, and Newman. [119]The Mel scale adapts frequencies as they are perceived by the human ear; consequently, it reflects the fact that the human ear distinguishes low-frequency differences better than high-frequency ones. [119] The Mel scale has a reference point at 1000 Hz, which corresponds to 1000 mels. Below 1000 Hz, the relationship between Hz and mels is approximately linear, while above it becomes logarithmic, according to the following equation and the trend shown in the Figure 73:

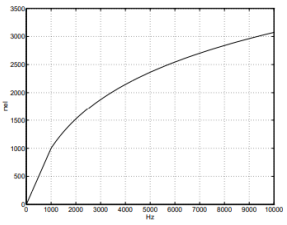

Figure 73: Conversion from frequency in Hz to the Mel scale.[120]

$$\text{mel}(f) = \begin{cases} f & f \leq 1000Hz \\ 2595 * log_{10}\left(1 + \frac{f}{700}\right) & f > 1000Hz \end{cases} \quad (46)$$

More specifically, the Mel-scaled spectrogram is obtained by applying a Mel filter bank to the power of the linear spectrogram. The filter bank consists of triangular filters centered at frequencies equally spaced on the Mel scale, with exponentially increasing bandwidths, in order to aggregate the energy of nearby bands and better reproduce human auditory perception.[120] In practice, a FFT with a window size of 1024 samples and a hop length of 512 samples was used to increase model fidelity. Moreover, 128 Mel filters were applied. The implementation was carried out in PyTorch using the MelSpectrogram() function from the torchaudio.transforms library.

   - Log-Mel Spectrogram: this consists of applying a logarithmic transformation to the Mel-scaled spectrogram. This operation further reflects human auditory

perception, since the human ear perceives sound intensity in a logarithmic manner. [119]

The logarithm of the previously obtained Mel spectrogram was computed, with the addition of a constant term (1×10^{-9}) to avoid $-\infty$ values corresponding to zeros:

$$\text{LogMelSpec} = \log(\text{MelSpec} + 1\text{x}10^{-9}) \quad (47)$$

- MFCC: These represent features extracted from the log-Mel spectrogram. After applying the log-Mel spectrogram, the Mel bands are still correlated with each other; therefore, a decorrelation is applied to each frame using the Discrete Cosine Transform (DCT), which allows obtaining a representation called the cepstrum.[121] The formula for the DCT is as follows:

$$c_k = \sum_{n=1}^{N} Y_n \cos[k\frac{(n-1/2)\pi}{N}], k = 0, \dots, K \quad (48)$$

where c_k is the k-th MFCC, K is the number of MFCC coefficients to obtain, $Y_n$is the log-Mel spectrogram, and N is the number of filters in the Mel filterbank.[120]

The cepstrum separates the periodic variation of the signal (pitch) from the slow variation (timbre), focusing on the latter, which contains most of the information useful for speech recognition.[121]

The final output consists of MFCC coefficients organized along the time axis, thus representing features extracted from the signal as a function of time and compactly capturing the main characteristics of the sound, supporting tasks such as speech recognition and speaker identification. [121]

Specifically, 13 MFCCs were extracted per frame. To the original coefficients, the first-order derivatives (delta), which indicate the rate of change of the MFCC coefficients, and the second-order derivatives (delta-delta), which indicate the acceleration or change in the rate of the coefficients, were added. These derivatives are concatenated with the original MFCC coefficients, resulting in a final dimension of 39 features per frame. The resulting MFCC matrix therefore has dimensions [time_steps × 39], where time_steps depends on the duration of the audio segment.

The procedure and the values used are based on the papers "Speech task based automatic classification of ALS and Parkinson's Disease and their severity using log Mel spectrograms" [101] and "Delta-Spectral Cepstral Coefficients For Robust Speech Recognition" [122]. The MFCCs were computed using the MFCC() function from the torchaudio.transforms library.

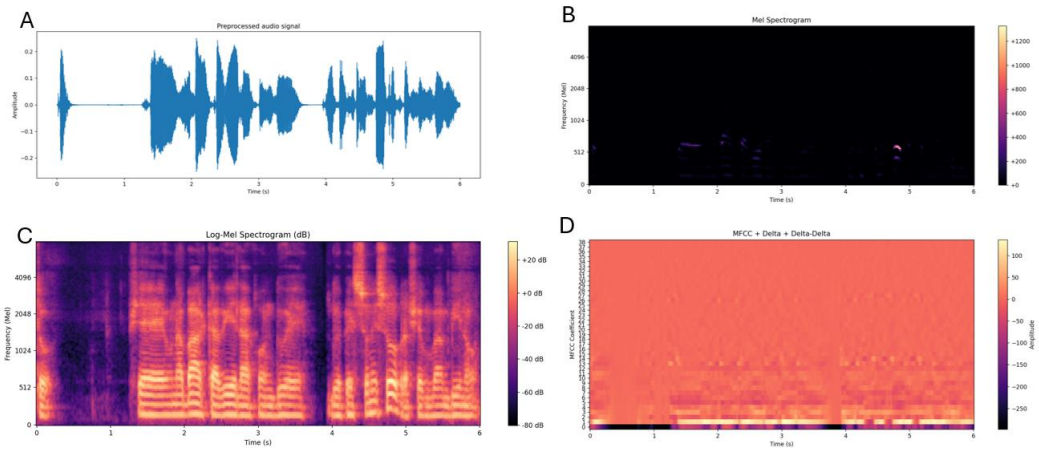

Figure 74: Graphical representation. (A) Preprocessed original audio signal, (B) Mel Spectrogram, (C) Log-Mel Spectrogram, and (D) MFCC.

6. **Data Storage and Management in HDF5 Format**
   Each audio fragment was saved in a final dataset in HDF5 format. To store all the necessary information, the dataset class was initially modified to save the name of each audio fragment followed by its fragment index, in order to preserve the correct order. Subsequently, various attributes were added in detail: the audio, the Mel spectrogram, the log-Mel spectrogram, the MFCCs, class label, and the patient's name. An illustration of the HDF5 architecture is shown in Figure 75:.

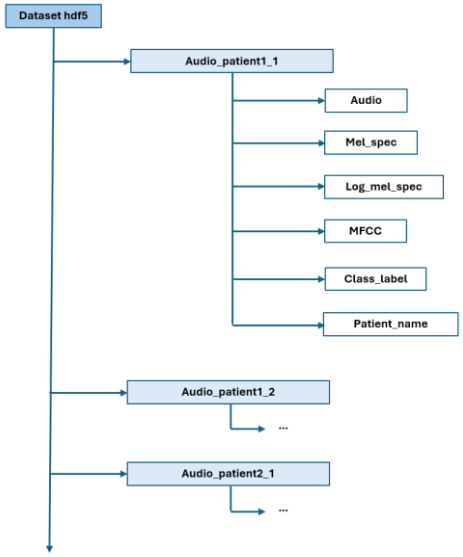

Figure 75: Structure of the HDF5 dataset used in this study.

5.4.2 Network Architectures for Audio Analysis

The initial idea of which networks to explore for processing the audio recordings of PPA patients was inspired by the work of Daniel Escobar-Grisales at al.[97], which investigates both a 2D CNN approach and a Wav2Vec approach. Our implementations were based on this concept and subsequently refined through extensive

experimentation. Three types of networks were developed: 2D-CNN, 2D-CNN combined with a biLSTM, and Wav2Vec.

5.4.2.1 2D-CNN Architecture

The Conv2DNet model was implemented using the PyTorch library in Python. The input to the model corresponds to the spectrograms described previously, provided in batches of data, with batch sizes ranging from 32 to 64 depending on the length of the chosen spectrogram. The network architecture is organized into three convolutional blocks:

- First convolutional block: a 2D convolutional layer with 64 filters, a 3×3 kernel capable of capturing local details, with padding of 1 to preserve spatial dimensions and include border pixels, and a stride of 1.
- Second convolutional block: a convolutional layer with 128 filters, a 3×3 kernel, and padding = 1.
- Third convolutional block: a convolutional layer with 192 filters and a 3×3 kernel. The increase in filters allows the extraction of increasingly complex features.

After each convolutional layer, a 2D batch normalization operation is applied, which stabilizes the network, reduces sensitivity to the initial weight values, and prevents exploding or vanishing gradient phenomena. This is followed by a ReLU activation function, used to introduce non-linearity into the network. To reduce the spatial dimensions of the feature maps, a 2×2 max pooling layer with a stride of 2 is applied, producing more compact and robust representations of the data

The final feature maps after the last pooling layer are flattened using a Flatten layer before passing to the fully connected layers. The first fully connected layer consists of 1024 neurons and is followed by a ReLU activation. The network also includes a dropout layer with a probability of 0.3, which randomly deactivates some neurons during training to prevent the model from relying excessively on specific features and to encourage the development of more robust representations. During prediction, the dropout is deactivated so that all connections are active. The final fully connected layer outputs logits, which are raw scores for each class. The predicted class is determined by the index of the maximum logit.

Since the dataset was highly imbalanced, class weights were assigned during training to give more importance to underrepresented classes. The loss function used is the weighted cross-entropy loss based on class frequency, and the optimizer employed is Adam, with a selected learning rate of 10^{-7}.

The model was trained for 250 epochs, with several strategies to improve generalization:

- A ReduceLROnPlateau scheduler was applied to the learning rate, based on the validation loss. If the loss does not improve for 10 consecutive epochs, the learning rate is halved. This helps the model escape local minima.
- The best model is saved according to the lowest validation loss.

- To prevent overfitting, an early stopping criterion based on the validation loss was implemented: if no improvement is observed for 50 consecutive epochs, training is stopped.

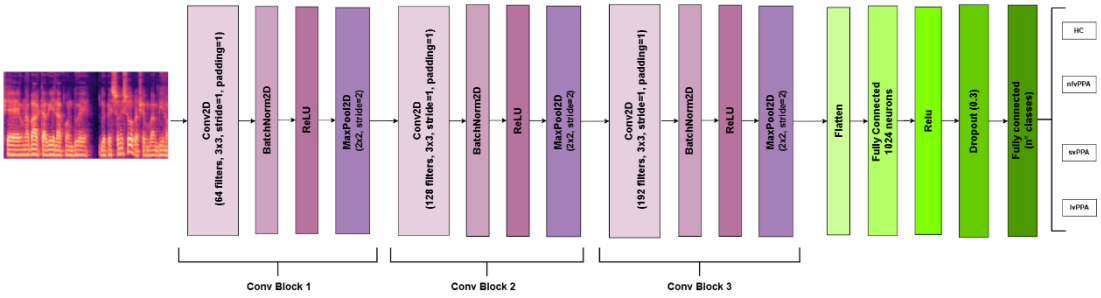

Figure 76: Illustration of the 2D-CNN Architecture.

5.4.2.2 2D-CNN + biLSTM Architecture

To test a slightly different approach, a model was implemented starting from the initial 2D CNN configuration, incorporating a BiLSTM layer. The main operations that allowed the model to be built are listed below.

The model employs the three previously described two-dimensional convolutional blocks to extract spatial features. The output of the three convolutional layers has dimensions [batch, channels, frequency, time].

Subsequently, to exploit temporal information with an LSTM layer, the 2D feature maps are transformed into temporal sequences. The CNN output is permuted and reshaped into [batch, timestep, features], where each timestep corresponds to a column along the time axis, and the features are the concatenation of channels and frequencies. To prevent the feature vector for each timestep from being too large and making the LSTM computationally heavy, a linear layer is applied to reduce the features to 512 elements per timestep.

The resulting sequences are passed through a bidirectional BiLSTM with 512 hidden units per direction (1024 units in total). This allows capturing temporal information in both directions, improving the representation of features along the temporal sequence. After the BiLSTM, a global max pooling operation is applied along the time axis, resulting in a vector of size [batch, 1024]. In this way, the most relevant information for each feature is preserved along the entire sequence, avoiding the use of a flatten layer and reducing the risk of overfitting.

The vector is then passed through a first fully connected layer that reduces the features by half, followed by a ReLU activation function and a dropout of 0.4 for regularization. Finally, a second fully connected layer produces the output logits for each class. The final class is selected as the one corresponding to the highest logit.

The model was trained with a batch size of 64 or 32 (depending on the input dimensions of the spectrogram extracted from the audio fragments), for 1500 epochs using a learning rate of 10^-7. All other settings, including those related to the optimizer, scheduler, and weights, remained identical to the previous configuration.

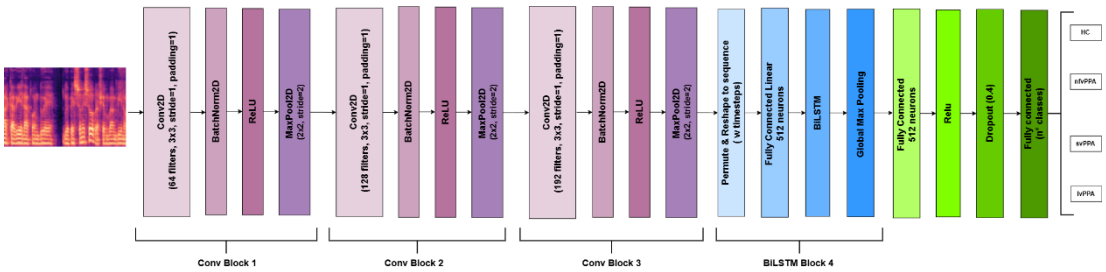

Figure 77: Illustration of the 2D-CNN + BiLSTM Architecture.

5.4.2.3 Wav2vec2.0 Architecture

An additional approach involved the use of a pre-trained model based on the Wav2Vec2 architecture. The model used is the pre-trained XLSR-Wav2Vec2 Large Italian version (jonatasgrosman/wav2vec2-large-xlsr-53-italian[123]), publicly available on *Hugging Face*, and was subsequently fine-tuned for binary and multi-class classification of speech recordings from HC and PPA subjects.

The Wav2Vec2 model takes as input the raw 1D audio signal and does not require transformation into a spectrogram, unlike the previously applied CNNs. The model architecture consists of a structure common to all Wav2Vec2 models, composed of:

- Feature extractor: a series of 7 convolutional layers designed to capture local acoustic patterns in the audio signal. These layers learn low-level features such as phonetic and spectral characteristics, producing a compressed representation of the input waveform.
- Contextualizer: a stack of 24 Transformer layers that model temporal relationships between acoustic frames and produce contextualized representations. The output of the Transformers is a sequence of dimension [time, features].

Subsequently, to obtain a compact representation of the features to be fed into the fully connected layer, a global max pooling operation along the time axis is applied. The vector resulting from the max pooling represents a compact embedding of the entire audio, incorporating all the relevant information. This flattened vector is then passed to a classification head, which aims to associate a class to the input vector. The classification head is composed of:

- A dropout layer, corresponding to the dropout pre-set in the Wav2Vec2 model
- A fully connected layer with a tanh activation function
- A second dropout layer to prevent overfitting
- A final fully connected layer that produces the logits, which are then used to assign the class.

Initially, the Wav2Vec2.0 model was kept completely frozen, training exclusively the classification head for three epochs.

Subsequently, gradual fine-tuning was performed by unfreezing the last two Transformer layers of the model, which were trained together with the classification head for a total of 50 epochs.
For model optimization, a batch size of 16 was used. The loss function employed was CrossEntropyLoss, and the optimizer used was AdamW, a variant of Adam with weight decay management set to 0.01, useful for regularizing the weights and reducing the risk of overfitting. The learning rate adopted for training was $1x10^{-5}$, a value that allows stable weight updates during training.
The final model was selected based on the highest balanced accuracy on the validation set, thus ensuring balanced performance across classes.

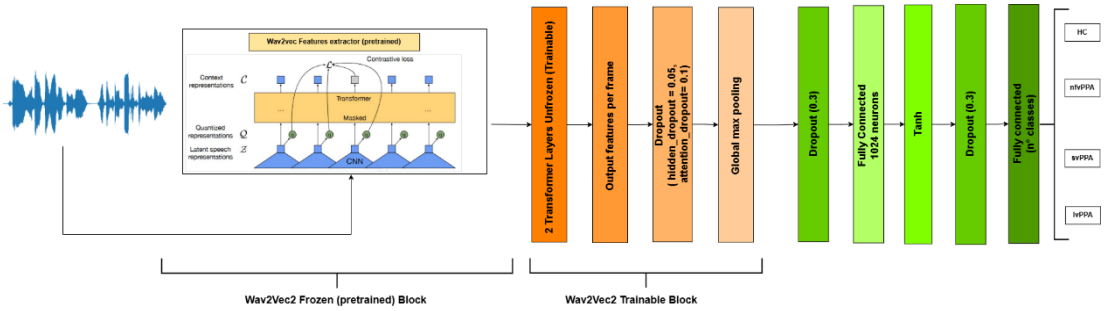

Figure 78: Illustration of the Wav2vec2.0 Architecture.

5.4.3 Weighted Error Penalization Strategy

Regarding the 2D-CNN and 2D-CNN + BiLSTM models, during training, a strategy of error penalization was applied to further improve performance. Instead of treating all errors equally, different weights are assigned depending on the severity of the error made by the network. More severe errors are penalized more heavily, while less critical errors receive lower penalties.
The weighted matrices were determined based on the following assumptions:

- Misclassifying an HC subject as PPA is less severe than the opposite case
- Confusing the non-fluent variant with the logopenic variant and vice versa is less severe
- Confusing the semantic variant with either of the other two variants is more severe.

In each matrix, the element (i,j) indicates the penalty associated with classifying a sample of class i as belonging to class j. At the implementation level, the standard CrossEntropyLoss function is transformed into a weighted cross-entropy loss, where each error is multiplied by the corresponding weight in the matrix. This approach drives the network's optimization towards minimizing the most severe errors.

5.4.4 Data Augmentation for Audio

The presence of a small number of subjects led us to apply data augmentation techniques. These techniques were used during data loading through the PyTorch *DataLoader* function, starting from data stored in HDF5 files.

In particular, two main strategies were adopted, both taken from the scientific literature.

The first strategy operates at the spectrogram level, as described in the article of Daniel S. Park at al.[124] The article proposes three types of data augmentation: linear time warping, time masking and frequency masking.

- Linear Time Warping: This operation deforms the spectrogram along the time axis. In this work, the deformation was carried out using bilinear interpolation in PyTorch, applying a random temporal scale between 0.95 and 1.02. If the spectrogram is stretched, the exceeding part is subsequently trimmed; if it is compressed, zero-padding is applied to maintain the original size. In this way, a temporal variant of the signal is generated.
- Frequency Masking: In this technique, f consecutive mel frequency channels are masked. The value of f is randomly chosen within the range 1–5, and the masked strip is positioned at a random point along the frequency axis, from the beginning up to the maximum frequency of the spectrogram. The operation was implemented using the *T.SpecAugment* function from *Torchaudio*. This technique simulates the loss or variation of spectral information.
- Time Masking: In this technique, t consecutive time steps are masked, where t is randomly chosen up to a maximum of 10 steps. The starting point of the masking is selected along the entire duration of the spectrogram. This operation was also carried out using the *T.SpecAugment* function from *Torchaudio*. In practice, temporal masking simulates pauses or interruptions in the signal.

All three techniques were applied with a probability of 50% for each patient, randomly, ensuring that the same operation was applied to all fragments of the same patient. In this way, multiple versions of the data were generated without altering the main characteristics of the signals. This configuration is reported in the following subsection *Audio Network Preprocessing Comparison* as **Test A**.

The second strategy is based on the approach proposed in the paper “Speech Emotion Recognition using Mel Spectrogram and Convolutional Neural Networks (CNN)” [125]. In this case, the strategy acts directly at the audio signal level, introducing modifications to increase the number of fragments. Two main techniques were applied: pitch shift and time stretch.

- Pitch shift: The pitch shift technique alters the original pitch of the sound, either increasing or decreasing it, while keeping the signal duration unchanged. The implementation was carried out using the *librosa.effects.pitch_shift* function, applying variations of +3 and −3 semitones.

- Time stretch: This technique alters the speed or duration of a sound: a factor of 0.8 was set to produce a slow time stretch, while a factor of 1.3 was set to generate a fast time stretch. This operation was performed using the *librosa.effects.time_stretch* function. Since these transformations modify the length of the fragments, it was necessary to intervene to maintain the original size: in cases of signal elongation, the fragment was trimmed, while in cases of shortening, zero-padding was applied.

This second strategy was initially applied to 50% of the patients, ensuring that the same technique was applied to all fragments of the same patient. This configuration is reported in the following subsection *Audio Network Preprocessing Comparison* as **Test B**. In a second phase, the same strategy was applied randomly to 50% of the audio fragments, independently of the patients, to further increase the variety of the data. This configuration is reported in the following subsection *Audio Network Preprocessing Comparison* as **Test C**.

5.4.5 Subject-Level Classification via Majority Voting

After training the three types of networks, each audio segment in the test set was assigned its predicted class. Using the subject's identifier, all segments belonging to the same subject were grouped together along with their predicted labels, and a majority voting strategy was applied. This strategy assigns to each subject the class that occurs most frequently among their audio segments. This post-processing method helps stabilize the model's performance, reduce the uncertainty caused by noisy or ambiguous segments, and ensures a coherent classification at the subject level.

5.4.6 Audio Network Preprocessing Comparison

In the analysis, several comparisons were conducted to determine the optimal preprocessing for each network, and the different tests performed are summarized in Table 5.

- **Test 1:** This comparison was carried out to identify which spectrogram (Mel, Log-Mel, or MFCC) was most suitable as input for the networks. The selection was performed on an initial CNN2D network, capable of classifying only two classes (HC vs PPA). The chosen configuration was then retained for the multiclass approach and for the network with the BiLSTM module.
- **Test 2:** This comparison was conducted to identify the most effective preprocessing strategies, focusing on different segment lengths, overlap durations, and normalization methods. Six approaches were tested:

- Segment of 6 s, overlap of 1s, normalization on the entire audio ("Pre").
- Segment of 6 s, overlap of 1s, normalization on each segment ("Post").
- Segment of 6 s, overlap of 3s, normalization on the entire audio ("Pre").
- Segment of 6 s, overlap of 3s, normalization on each segment ("Post").
- Segment of 12s, overlap of 6s, normalization on the entire audio ("Pre").
- Segment of 12s, overlap of 6s, normalization on each segment ("Post").

These tests were applied to the CNN2D, CNN2D + BiLSTM and Wav2vec2.0 networks. All subsequent preprocessing steps were carried out in the multiclass setting, with four classes: HC, nfvPPA, svPPA, and lvPPA, as this approach proved to be more sensitive to differences between segments compared to the initial binary setup (HC vs PPA). The goal was to select the configuration that achieved high performance on the validation set in terms of accuracy and F1-score, while also maintaining good generalization on the test set. Discrepancies between validation and test performance were penalized to favor stable and balanced configurations.

- **Test 3:** This test was applied to both the CNN2D and CNN2D + BiLSTM networks in the multiclass setting. The technique involved assigning different weights to errors to improve overall performance. Its effectiveness was evaluated by comparing results on both the validation and test sets, considering not only accuracy metrics but also the networks' generalization ability.
- **Test 4:** This test was applied to both the CNN2D and CNN2D + BiLSTM networks, in the multiclass settings. Different data augmentation strategies (Tests A, B, and C, as explained in the subsection *Data Augmentation for Audio*) were evaluated to improve performance. Network configurations were selected based on their results on the validation and test sets, taking into account balanced accuracy, F1 score, generalization ability, and training efficiency. Data augmentation was applied only when it provided a clear benefit without causing overfitting or significantly increasing training time. At the Wav2vec level, this approach was discarded due to the high computational cost required for each trial. Even without applying data augmentation, Wav2vec already demanded considerable computational resources.

Test	Type of Comparison	CNN2d	CNN2D +biLSTM	Wav2vec2.0
1	Selection of the best spectrogram among Mel Spectrogram, Log-Mel Spectrogram, and MFCC	✓	✗	✗
2	Application of different segment settings (6s overlap 3, 6s overlap 1, 12s overlap 6) and different normalization strategies (before or after segment creation) as network input.	✓	✓	✓
3	Application of regularization / penalty terms	✓	✓	✗
4	Data augmentation: Test A, Test B, Test C	✓	✓	✗

Table 5: Summary of Preprocessing Tests Applied to Audio Networks

5.5 MRI-Based Deep Learning Models

This section describes the preprocessing and the creation of the model, which, starting from 3D T1-weighted brain MRI images, is capable of distinguishing between HC, patients with PPA, and its variants. Although this approach has not previously been applied to PPA, it builds upon existing methods developed for 3D T1-weighted images in the study of brain atrophy in other pathologies.

5.5.1 3D T1 MRI Preprocessing Pipeline

MRI images of each subject were preprocessed as follows:

- Removal of the anatomical part of the neck to isolate the brain.
- Segmentation of brain tissues into grey matter, white matter, and cerebrospinal fluid (CSF) using Statistical Parametric Mapping (SPM, version 12) in MATLAB

(v.2023). During this step, SPM also produced a modulated version of each T1-weighted image by applying the Jacobian determinants derived from the affine component estimated during segmentation. Modulation was used to preserve the global volumes of the tissues, compensating for any changes introduced by the affine normalization.

- Removal of the scalp and skull using the Brain Extraction Tool (BET) in FSLeyes, version 5.0.10, obtaining a clean brain image.
- The complete, modulated, and BET-processed brain images were spatially normalized to the standard stereotactic space of the Montreal Neurological Institute (MNI, linear MNI152 T1 1mm template) using only affine transformations (translation, rotation, scaling, and shearing), with isotropic voxel resolution of 1 × 1 × 1 mm^3, avoiding non-linear deformations. This linear normalization aligns each subject's brain to a common stereotactic coordinate system while preserving individual anatomical geometry without introducing local deformations.
- The resulting modulated and affine-normalized T1-weighted images were subsequently reduced through a patch before being fed to the network. The patch is created to reduce the number of voxels, focusing on the areas most affected by atrophy in PPA subjects and avoiding providing confusing voxels to the network.[33] For this reason, the patch excludes all voxels belonging to the right hemisphere and a portion of the occipital region, areas generally less affected by atrophy. The final patch has a size of 79x150x160.

5.5.2 3D-CNN Architecture

This section describes the network developed to classify 3D T1-weighted image patches for each subject, distinguishing healthy controls (HC) from PPA patients and its variants. For the creation of the deep learning model, we started from a network previously published by S. Basaia at all.[92].

The 3D-CNN architecture was simplified since, working on patches, a large number of convolutional layers was not necessary. The final network used consists of six 3D convolutional layers and receives as input patches of size 79×150×160 voxels. The six convolutional layers and their characteristics are listed below:

- Layer 1: 44 feature maps, kernel 3×3×3, stride 1
- Layer 2: 44 feature maps, kernel 3×3×3, stride 2
- Layer 3: 88 feature maps, kernel 3×3×3, stride 1
- Layer 4: 88 feature maps, kernel 3×3×3, stride 2
- Layer 5: 176 feature maps, kernel 3×3×3, stride 1
- Layer 6: 176 feature maps, kernel 3×3×3, stride 2

The alternating stride of 1 and 2 is used to perform downsampling when set to 2, avoiding the need for a separate pooling layer. After each convolutional layer, a ReLU activation function is applied to introduce nonlinearity into the extracted features. The features are then flattened through a flatten layer before entering a fully connected layer with 1000 units, followed by a ReLU activation and a dropout of 30% to reduce overfitting. The final output is produced by a second fully connected layer that outputs the logits; the assigned class corresponds to the maximum logit value.
During training, the network uses a CrossEntropyLoss function, weighted according to the class distributions to compensate for class imbalance in the dataset. The weight optimizer is Adam, with a learning rate of 1 × 10^-7. The learning rate is managed by a ReduceLROnPlateau scheduler, which reduces the learning rate by a factor of 0.1 if the validation loss does not improve for 10 consecutive epochs. The model is trained for a maximum of 500 epochs, with a batch size of 4. Additionally, the code implements an early stopping mechanism in case of no improvement in the validation loss for 100 consecutive epochs.

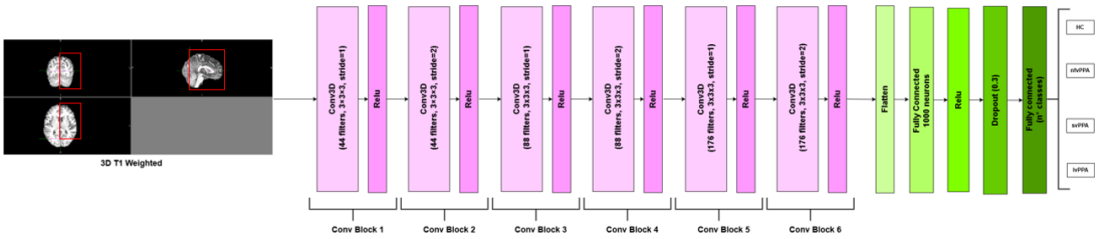

Figure 79: Illustration of the 3D-CNN Architecture.

5.5.3 Data Augmentation for MRI

A data augmentation strategy was applied to increase the number of input images to the network. The data augmentation was performed online, within the dataset, with a 50% probability of applying one of the following transformations:

- Elastic Transformation: Applies an elastic deformation to the image. This transformation was implemented using the *ElasticTransform* function from the *torchvision.transforms* module in *PyTorch*, with two adjustable parameters: α and σ. The parameter α controls the intensity of the deformation, while σ determines its smoothness and continuity. In this study, the selected parameters were α = 70.0 and σ = 7.0.[126]
- Random Rotation: Rotates the image by a random angle between –α and +α, with the selected α equal to 15°.This transformation was implemented using the *RandomRotation* function from the *torchvision.transforms* module in *PyTorch*.[127]

5.5.4 Train-Validation-Test Split and Cross-Validation

The dataset was divided into training, validation, and test sets. In the first phase of the project, the dataset was split into 80% of patients for training and 20% for testing. Subsequently, the training set was further divided into 80% for training and 20% for validation. The model was trained using the training set, while the validation set was used simultaneously to monitor the reduction of the loss function and prevent overfitting. The test set, on the other hand, was used to evaluate the trained model and assess its generalization capability.

For the audio-based networks, the splits were created with the precaution of assigning all fragments from a single patient to the same set, thereby avoiding the presence of data from the same patient in different sets. Subsequently, a class-stratified split was applied to ensure a similar balance of classes across the training, validation, and test sets.

For the 3D MRI image networks, since there is only one image per patient, this issue does not arise. A stratification technique was applied solely to maintain class balance. Stratification was implemented using the *train_test_split* function from the Python *scikit-learn* library, which simplifies this type of division.

To evaluate performance more robustly and reduce dependency on a single split, a hybrid nested cross-validation was adopted, dividing patients into three folds. In each iteration, one fold is used as the test set, while the remaining folds are used as the training set. The training set is then further split into training and validation sets according to the usual 80%-20% proportion. This process is repeated by rotating the folds between test and training.

This hybrid approach reduces the number of trainings required compared to a classical nested cross-validation, thereby lowering the computational cost.

5.6 Model Explainability

In terms of explainability, Grad-CAM was used to identify the areas on which the 2D CNN, applied to spectrograms, and the 3D CNN, applied to 3D T1 images, focused most. Grad-CAM can be applied to any CNN architecture and allows visualization of which regions of the image contribute most to the class prediction. The technique leverages gradients with respect to the output of interest to weight the feature maps of a convolutional layer.

The procedure generates a weighted activation map, to which a ReLU function is applied to retain only the features that contribute positively to the target class.[128] The

resulting map can then be overlaid on the image, enabling visualization of the most relevant areas.

At the wav2vec2.0 network level, an explainability approach based on Integrated Gradients (IG) was applied. Integrated Gradients is a technique that can be applied to any differentiable neural network and allows quantification of the contribution of each input feature to the network's prediction. The method compares the actual input with a baseline reference (e.g., an input with all zeros) and computes the integral of the gradients of the output with respect to the input along a straight-line path from the baseline to the actual input.

This procedure produces a feature attribution map, where each input element is assigned a score representing its contribution to the target class.[129]

5.7 Multimodal Integration: Combining Audio and MRI-based Models

Starting from the assumption that the main network to be developed was a multi-class model, capable of recognizing the four classes, HC, nfvPPA, svPPA and lvPPA, a multimodal late fusion approach was adopted to further enhance the performance of the networks based on both the Picnic Scene Test audio recordings and the 3D T1-weighted MRI images. For the audio modality, the network selected for fusion was chosen as the best-performing one among the three proposed models.

The two networks were trained on different datasets, since only a subset of subjects had both MRI and audio recordings available. For many participants, especially healthy controls, multimodal data were not available. Therefore, the audio model was trained on the entire set of available subjects, while the image network was trained on the complete MRI dataset, ensuring that the validation and test sets included subjects in common with those used for the audio network.

Subject-level predictions were computed separately for each network.

For the audio network, since a majority voting approach was applied over individual audio fragments, the class probabilities were obtained as the ratio between the number of fragments assigned to a given class and the total number of fragments available for that subject.

For the image network, the output of the 3D-CNN corresponded to the logits, i.e., real-valued numbers representing the network's confidence for each class, which were subsequently converted into class probabilities in the range [0,1] using the softmax function.

The multimodal fusion was implemented by combining the predictions from both networks through a weighted sum. Specifically, the fused probabilities were computed as:

$$p = w * p_{audio} + (1 - w) * p_{MRI} \quad (49)$$

Where p_{audio} and p_{MRI} represent the class probabilities predicted by the audio and MRI networks, respectively, and $w$is a weight optimized to maximize the balanced accuracy on the validation set. The optimal weight $w$was then applied to the test set, allowing the evaluation of the performance improvement achieved through multimodal fusion compared to the individual models.
This approach made it possible to integrate information derived from both vocal and imaging modalities, leading to an overall improvement in the network's classification performance.

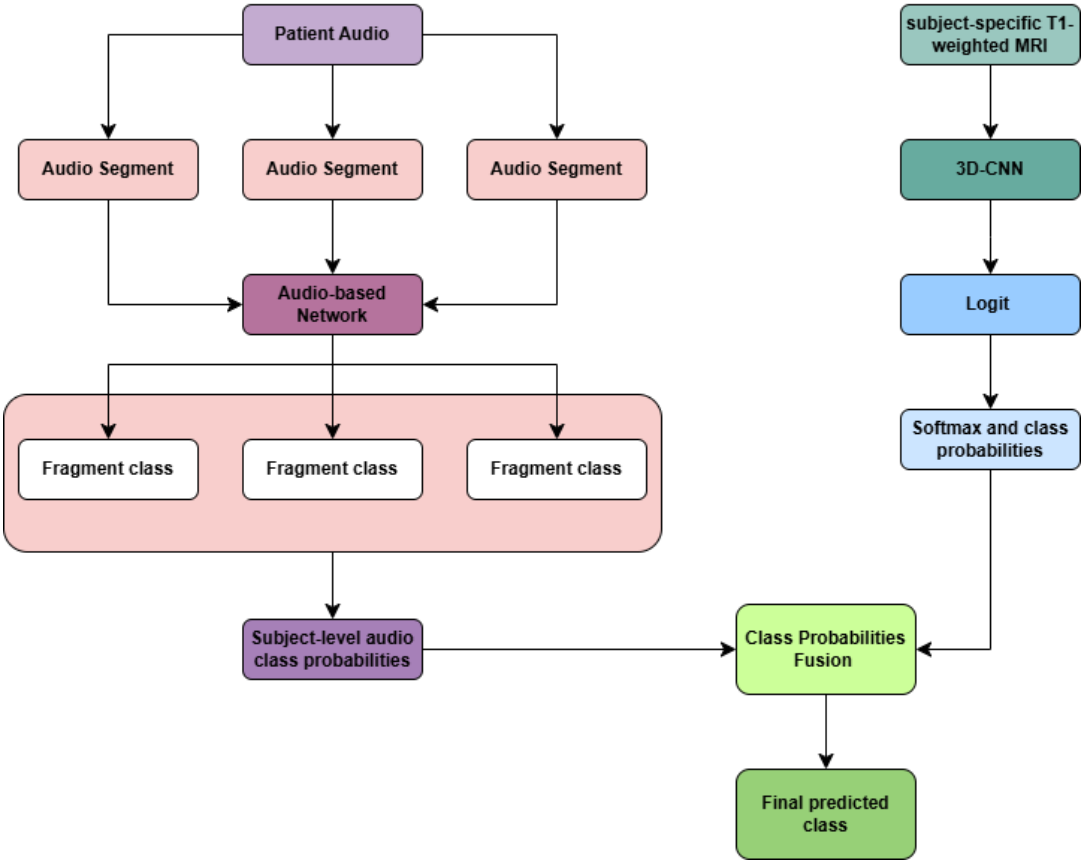

Figure 80: Visual representation of the late fusion process.

5.8 Overall comparisons

For all the networks created, several classification tasks were performed to evaluate their performance in different scenarios. All the tasks performed are summarized in Table 6.

Task	Type of Classification	Class compared
1	Multiclass	HC, svPPA, lvPPA, nfvPPA
2	Multiclass	svPPA, lvPPA, nfvPPA
3	Binary	HC, PPA
4	Binary	HC, nfvPPA
5	Binary	HC, svPPA
6	Binary	HC, lvPPA
7	Binary	nfvPPA, svPPA
8	Binary	lvPPA, nfvPPA
9	Binary	lvPPA, svPPA

Table 6: Overview of Tasks Performed for Each Network

The Table 7 shows Task 10, related to network fusion, used to evaluate the combined approach between the audio network and the 3D-CNN image-based network. To perform this procedure, the same validation and test sets were used for both the audio and image networks.

Task	Type of Classification	Class compared	Network
10	Multiclass	HC, svPPA lvPPA nfvPPA	Audio network + 3D-CNN

Table 7: Task for evaluating the fusion strategy.

6. Results

6.1 Participants

Table 8 and Table 9 provide a detailed overview of the sample population included in this thesis, reporting sample size, age at the time of examination, sex, years of education, and, for subjects with PPA, disease duration (in months). Specifically, *Table 8* presents a comparison between healthy controls and PPA patients (considered as a single group) matched for age, sex, and years of education. Table 9 shows comparisons among the different PPA subtypes (svPPA, nfvPPA, lvPPA). Patients with nfvPPA had fewer years of education and a shorter disease duration compared to svPPA, while lvPPA patients were significantly older than svPPA patients.

Group	N. participants	Age (years)	Sex		Years of education	Disease Duration (months)
			Female	Male		
HC	91	67.47± 5.45 (56.25-81.41)	48	43	11.80±4.17 (5-22)	-
PPA (Total)	94	68.88±8.46 (42.06-83.93)	55	39	11.04±4.70 (3-22)	36.71±19.11 (3.02-126.59)

Table 8: Demographic and clinical data of HC and PPA participants.

Group	N. participants	Age (years)	Sex		Years of education	Disease Duration (months)
			Female	Male		
nfvPPA	38	69.71 ±8.59 (51.58-83.93)	24	14	9.11±4.83 (3-22) *	28.55±13.15 (3.02-68.01) *
svPPA	36	66.23±8.49 (42.06-81.63)	20	16	12.42±4.42 (5-18)	45.76±21.90 (11.33-126.59)
lvPPA	20	72.09±6.93 (56.34-81.32) *	11	9	12.25±3.77 (5-17)	35.91±16.70 (5.59-62.06)

*Table 9: Demographic and clinical data of PPA participants. * = statistically significant difference with svPPA*

6.2 Audio-Based Model Results

This section presents the results of the various tests reported in Table 5.

6.2.1 Test 1: Spectrogram Selection

Table 10 shows the performance obtained in Test 1, applied to the CNN2D network trained in a binary comparison between HC subjects and PPA subjects, using different types of input spectrograms: Mel spectrogram, Log-Mel spectrogram, and MFCC. The Log-Mel spectrograms achieved the best results, with a validation accuracy of 0.828, and a test accuracy of 0.892.

Test 1	Balanced accuracy Validation	F1 score Validation (Weighed)	Balanced accuracy Test	F1 score Test (Weighed)
Mel -spectrogram	0.781	0.789	0.871	0.885
Log-Mel spectrogram	**0.828**	**0.837**	**0.892**	**0.914**
MFCC	0.653	0.709	0.660	0.756

Table 10: Comparison between Mel-spectrogram, log-Mel spectrogram, and MFCC as inputs to the CNN2D network

6.2.2 Test 2: Segment and Normalization Settings for Network Input

In tables Table 11, Table 12, Table 13 the results obtained in Test 2 are reported. This test concerns the selection of the most suitable audio preprocessing strategy for, respectively, the CNN2D, CNN2D + BiLSTM, and Wav2vec 2.0 networks. The tables present the different configurations evaluated, varying in window length, overlap length, and normalization strategy (before or after segmentation), together with the corresponding performance metrics on the validation and test sets.

The results of the 2D-CNN are reported in Table 11. The optimal preprocessing pipeline consists of normalizing the entire audio file and then segmenting it into 12-second windows with a 6-second overlap. This configuration achieved a balanced accuracy of 0.535 on the validation set and 0.493 on the test set.

Segments Length	Overlap Length	Normalization		Balanced accuracy Validation	F1 score Validation (Weighed)	Balanced accuracy Test	F1 score Test (Weighed)
		Pre	Post				
6s	1s	X		0.5011	0.505	0.505	0.508
6s	1s		X	0.488	0.489	0.445	0.446
6s	3s	X		0.485	0.486	0.485	0.487
6s	3s		X	0.523	0.527	0.485	0.483
12s	**6s**	**X**		**0.535**	**0.535**	**0.493**	**0.497**
12s	6s		X	0.535	0.534	0.461	0.468

Table 11: Comparison of preprocessing strategies for the 2D-CNN network.

The results of the 2D-CNN + BiLSTM, are reported in the Table 12. Among the configurations evaluated, the one using 6-second segments with a 3-second overlap and normalization applied after segmentation (fragment-level normalization) was selected. This configuration achieved a balanced accuracy of 0.533 on the validation set and 0.519 on the test set.

Although the 12-second segments reached very high accuracy values on the validation set, they showed a lower generalization ability on the test set, suggesting a possible overfitting on the validation data. In this case, shorter fragments were preferred, also considering the presence of the BiLSTM layer, which tends to perform better with shorter input sequences, reducing the risk of inefficient memory and improving learning in the temporal domain.

Segments Length	Overlap Length	Normalization		Balanced accuracy Validation	F1 score Validation (Weighted)	Balanced accuracy Test	F1 score Test (Weighted)
		Pre	Post				
6s	1s	X		0.529	0.525	0.514	0.513
6s	1s		X	0.499	0.495	0.535	0.531
6s	3s	X		0.527	0.520	0.492	0.505
6s	**3s**		**X**	**0.533**	**0.524**	**0.519**	**0.505**
12s	6s	X		0.604	0.596	0.533	0.517
12s	6s		X	0.640	0.630	0.504	0.503

Table 12: Comparison of preprocessing strategies for the 2D-CNN with BiLSTM network.

The results of the Wav2Vec 2.0, is shown in Table 10. The best configuration was found to involve segmentation into 12-second windows with a 6-second overlap,

followed by normalization. This configuration achieved a balanced accuracy of 0.513 on the validation set and 0.569 on the test set.

Segments Length	Overlap Length	Normalization		Balanced accuracy Validation	F1 score Validation (Weighted)	Balanced accuracy Test	F1 score Test (Weighted)
		Pre	Post				
6s	1s	X		0.454	0.451	0.527	0.510
6s	1s		X	0.475	0.472	0.505	0.480
6s	3s	X		0.482	0.477	0.540	0.530
6s	3s		X	0.482	0.483	0.534	0.530
12s	6s	X		0.513	0.464	0.571	0.560
12s	**6s**		**X**	**0.513**	**0.495**	**0.569**	**0.560**

Table 13: Comparison of preprocessing strategies for the Wav2vec2.0 network.

6.2.3 Test 3: Penalties

The results of Test 3 are shown in Table 14 and Table 15. This test evaluates the effect of applying a penalty to the model using the best audio input configuration identified in Test 2. The tables also include a row labeled “Original”, which reports the best performance obtained without applying any penalty.

Table 14 shows the results of applying the penalty to the CNN2D model. The best configuration remains the original one, with a balanced accuracy of 0.535 on the validation set and 0.493 on the test set. Although the penalty slightly increased balanced accuracy on the validation set, it did not improve performance on the test set, suggesting possible overfitting on the validation data.

2D-CNN	Balanced accuracy Validation	F1 score Validation (Weighted)	Balanced accuracy Test	F1 score Test (Weighted)
Test 3	0.564	0.495	0.494	0.498
Original	**0.535**	**0.535**	**0.493**	**0.497**

Table 14:2D-CNN Performance: With vs. Without Penalty

Table 15 shows the results of applying the penalty to the CNN2D + BiLSTM model. For this network as well, the original configuration was preferred, as it provides stable and consistent performance across both validation and test sets, with a balanced accuracy of 0.533 on the validation set and 0.519 on the test set.

2D-CNN + biLSTM	Balanced accuracy Validation	F1 score Validation (Weighted)	Balanced accuracy Test	F1 score Test (Weighted)
Test 3	0.514	0.521	0.622	0.6074
Original	**0.533**	**0.524**	**0.519**	**0.505**

Table 15: 2D-CNN + biLSTM Performance: With vs. Without Penalty

6.2.4 Test 4: Data Augmentation

The application of Test 4 involves the use of Data Augmentation, specifically through three configurations: Test A, Test B, and Test C. For each configuration, the resulting increase in the number of training audio files is summarized in Table 16.

	Number of original audio files	Number of augmented audio files
Test A	3329	1517
Test B	3329	1517
Test C	3329	1617

Table 16: Number of Original and Augmented Audio Files for Each Data Augmentation Method.

The results of Test 4 are shown in tables Table 17 and Table 18. This test evaluates different data augmentation approaches applied to both the 2D-CNN network and the 2D-CNN + BiLSTM network. The tables also include a row labeled "Original," which reports the performance obtained without applying data augmentation.
Table 17 shows the results for the 2D-CNN network. The original configuration, without data augmentation, was selected as the best setting because it offers an optimal compromise between performance and training time. The performance obtained with data augmentation on the validation and test sets is comparable to the original configuration, while the average training time per epoch is significantly lower for the original (18.2 s), especially compared to Tests B and C (35.8 s and 35.4 s, respectively).

2D-CNN	Balanced accuracy Validation	F1 score Validation (Weighted)	Balanced accuracy Test	F1 score Test (Weighted)	Average Training Time per Epoch
Test A	0.533	0.476	0.478	0.480	22.3s
Test B	0.527	0.521	0.485	0.490	35.8s
Test C	0.544	0.463	0.467	0.530	35.4s
Original	**0.535**	**0.535**	**0.493**	**0.497**	**18.2s**

Table 17: 2D-CNN Performance with Different Data Augmentation Strategies.

The same procedure was applied to the 2D CNN + BiLSTM network in the Table 18. Among the different configurations tested, Test B shows lower performance compared

to the original configuration, while Tests C and D show signs of overfitting, with very high validation accuracy (0.64 for Test C and 0.633 for Test D) but much lower test set accuracy (0.455 for Test C and 0.494 for Test D). Therefore, the original configuration remains the best choice, providing a balanced trade-off between performance, generalization, and training efficiency.

2D-CNN + biLSTM	Balanced accuracy Validation	F1 score Validation (Weighted)	Balanced accuracy Test	F1 score Test (Weighted)	Average Training Time per Epoch
Test B	0.4935	0.491	0.479	0.479	20.0 s
Test C	0.640	0.626	0.455	0.522	31.9s
Test D	0,633	0.621	0.494	0.501	39.1 s
Original	**0.533**	**0.524**	**0.519**	**0.505**	**19.0s**

Table 18: 2D-CNN + biLSTM performance with Data Augmentation Strategies.

6.2.5 Audio Network Overall Comparison

In this section, the results of CNN2D, CNN2D + BiLSTM, and Wav2Vec 2.0 are reported for all the tasks shown in Table 6. Model performance is presented both at the fragment level on the test set and at the patient level, after applying majority voting.

6.2.5.1 Multiclass Classification: HC, nfvPPA, svPPA, lvPPA

Table 19 shows the performance of the models on Task 1.

Test set	2D-CNN (12s, overlap 6s, normalization pre)	2D-CNN + biLSTM (6s, overlap 3s, normalization post)	Wav2vec2.0 (12s, overlap 6s, normalization post)
Accuracy	0.493	0.506	0.567
Balanced Accuracy	0.493	0.519	0.569
Precision (Macro)	0.505	0.516	0.553
F1-score (Macro)	0.497	0.513	0.569
F1-score (Weighted)	0.497	0.505	0.561
Accuracy Majority Voting	0.591	0.636	0.773
Balanced Accuracy Majority Voting	0.569	0.624	0.745
Precision Majority Voting (Macro)	0.629	0.618	0.767
F1-score Majority Voting (Macro)	0.568	0.610	0.745
F1-score Majority Voting (Weighted)	0.598	0.645	0.772

Table 19: Test-level comparison of the three main network models, 2D-CNN, 2D-CNN + BiLSTM, and Wav2Vec 2.0, for 4 class classification (HC, nfvPPA, svPPA and lvPPA).

6.2.5.2 Multiclass Classification: All PPA Variant

The following Table 20 shows the performance of the models on Task 2.

Test set	2D-CNN (12s, overlap 6s, normalization pre)	2D-CNN + biLSTM (6s, overlap 3s, normalization post)	Wav2vec2.0 (12s, overlap 6s, normalization post)
Accuracy	0.491	0.576	0.560
Balanced Accuracy	0.447	0.545	0.468
Precision (Macro)	0.444	0.581	0.390
F1-score (Macro)	0.447	0.545	0.410
F1-score (Weighted)	0.472	0.554	0.500
Accuracy Majority Voting	0.438	0.624	0.560
Balanced Accuracy Majority Voting	0.349	0.524	0.468
Precision Majority Voting (Macro)	0.407	0.448	0.390
F1-score Majority Voting (Macro)	0.349	0.524	0.410
F1-score Majority Voting (Weighted)	0.440	0.556	0.500

Table 20: Test-level comparison of the three main network models, 2D-CNN, 2D-CNN + BiLSTM, and Wav2Vec 2.0, for 3 class classification (nfvPPA, svPPA and lvPPA).

6.2.5.3 Binary Classification of HC vs PPA: Overall and Variant-specific Analysis

In this section, the results of Tasks 3,4,5, and 6, which correspond to binary comparisons between HC and PPA, are presented. Specifically, Task 3 is shown in Table 21, Task 4 in Table 22, Task 5 in Table 23, Task 6 in Table 24. For the CNN2D and CNN2D + biLSTM networks, a nested hybrid cross-validation approach was applied.

Test set Task 3: HC vs PPA	2D-CNN (12s, overlap 6s, normalization pre)	2D-CNN + biLSTM (6s, overlap 3s, normalization post)	Wav2vec2.0 (12s, overlap 6s, normalization post)
Accuracy	0.827 ± 0.042	0.842 ± 0.065	0.871
Balanced Accuracy	0.808 ± 0.032	0.802 ± 0.043	0.691
Precision (Macro)	0.764 ± 0.060	0.789 ± 0.086	0.763
F1-score (Macro)	0.775 ± 0.051	0.787 ± 0.072	0.718
F1-score (Weighted)	0.836 ± 0.036	0.847 ± 0.055	0.861
Accuracy Majority Voting	0.857 ± 0.030	0.849 ± 0.053	0.833
Balanced Accuracy Majority Voting	0.853 ± 0.030	0.819 ± 0.034	0.714
Precision Majority Voting (Macro)	0.837 ± 0.033	0.844 ± 0.067	0.905
F1-score Majority Voting (Macro)	0.841 ± 0.031	0.826 ± 0.050	0.747
F1-score Majority Voting (Weighted)	0.859 ± 0.029	0.848 ± 0.049	0.809

Table 21: Test-level comparison of the three main network models, 2D-CNN, 2D-CNN + BiLSTM, and Wav2Vec 2.0, for 2 class classification (HC and PPA).

Test set Task 4: HC vs nfvPPA	2D-CNN (12s, overlap 6s, normalization pre)	2D-CNN + biLSTM (6s, overlap 3s, normalization post)	Wav2vec2.0 (12s, overlap 6s, normalization post)
Accuracy	0.843 ± 0.044	0.817 ± 0.060	0.878
Balanced Accuracy	0.841 ± 0.043	0.807 ± 0.056	0.864
Precision (Macro)	0.838 ± 0.042	0.817 ± 0.059	0.878
F1-score (Macro)	0.839 ± 0.043	0.810 ± 0.056	0.871
F1-score (Weighted)	0.843 ± 0.044	0.815 ± 0.060	0.877
Accuracy Majority Voting	0.903 ± 0.071	0.847 ± 0.071	0.933
Balanced Accuracy Majority Voting	0.903 ± 0.071	0.848 ± 0.072	0.938
Precision Majority Voting (Macro)	0.902 ± 0.071	0.851 ± 0.076	0.938
F1-score Majority Voting (Macro)	0.902 ± 0.071	0.846 ± 0.071	0.933
F1-score Majority Voting (Weighted)	0.903 ± 0.071	0.847 ± 0.071	0.933

Table 22: Test-level comparison of the three main network models, 2D-CNN, 2D-CNN + BiLSTM, and Wav2Vec 2.0, for 2 class classification (HC and nfvPPA).

Test set Task 5: HC vs svPPA	2D-CNN (12s, overlap 6s, normalization pre)	2D-CNN + biLSTM (6s, overlap 3s, normalization post)	Wav2vec2.0 (12s, overlap 6s, normalization post)
Accuracy	0.878 ± 0.040	0.870 ± 0.046	0.633
Balanced Accuracy	0.878 ± 0.039	0.871 ± 0.046	0.627
Precision (Macro)	0.879 ± 0.040	0.870 ± 0.046	0.620
F1-score (Macro)	0.878 ± 0.040	0.870 ± 0.046	0.620
F1-score (Weighted)	0.878 ± 0.040	0.870 ± 0.046	0.638
Accuracy Majority Voting	0.939 ± 0.057	0.923 ± 0.057	0.615
Balanced Accuracy Majority Voting	0.936 ± 0.063	0.923 ± 0.056	0.613
Precision Majority Voting (Macro)	0.939 ± 0.055	0.920 ± 0.059	0.607
F1-score Majority Voting (Macro)	0.936 ± 0.060	0.921 ± 0.058	0.606
F1-score Majority Voting (Weighted)	0.938 ± 0.057	0.923 ± 0.057	0.620

Table 23: Test-level comparison of the three main network models, 2D-CNN, 2D-CNN + BiLSTM, and Wav2Vec 2.0, for 2 class classification (HC and svPPA).

Test set Task 6: HC vs lvPPA	2D-CNN (12s, overlap 6s, normalization pre)	2D-CNN + biLSTM (6s, overlap 3s, normalization post)	Wav2vec2.0 (12s, overlap 6s, normalization post)
Accuracy	0.824 ± 0.065	0.738 ± 0.103	0.848
Balanced Accuracy	0.824 ± 0.065	0.735 ± 0.107	0.865
Precision (Macro)	0.827 ± 0.062	0.763 ± 0.083	0.852
F1-score (Macro)	0.823 ± 0.066	0.722 ± 0.121	0.847
F1-score (Weighted)	0.823 ± 0.066	0.723 ± 0.119	0.849
Accuracy Majority Voting	0.862 ± 0.065	0.743 ± 0.108	0.917
Balanced Accuracy Majority Voting	0.828 ± 0.084	0.655 ± 0.137	0.938
Precision Majority Voting (Macro)	0.891 ± 0.055	0.638 ± 0.249	0.900
F1-score Majority Voting (Macro)	0.835 ± 0.085	0.631 ± 0.192	0.911
F1-score Majority Voting (Weighted)	0.853 ± 0.075	0.690 ± 0.154	0.919

Table 24: Test-level comparison of the three main network models, 2D-CNN, 2D-CNN + BiLSTM, and Wav2Vec 2.0, for 2 class classification (HC and lvPPA).

6.2.5.4 Binary Classification Between PPA Variants

In this section, the results for Task 7, 8, and 9, which correspond to comparisons between the models in classifying PPA variant, are presented. Table 25 shows Task 7, Table 27 shows Task 8 and **Errore. L'origine riferimento non è stata trovata.** shows Task 9. For the CNN2D and CNN2D + biLSTM networks, a nested hybrid cross-validation approach was applied.

Test set Task 7 nfvPPA vs svPPA	2D-CNN (12s, overlap 6s, normalization pre)	2D-CNN + biLSTM (6s, overlap 3s, normalization post)	Wav2vec2.0 (12s, overlap 6s, normalization post)
Accuracy	0.617 ± 0.067	0.603 ± 0.064	0.630
Balanced Accuracy	0.617 ± 0.066	0.626 ± 0.048	0.632
Precision (Macro)	0.625 ± 0.073	0.649 ± 0.021	0.664
F1-score (Macro)	0.611 ± 0.063	0.587 ± 0.084	0.611
F1-score (Weighted)	0.614 ± 0.063	0.580 ± 0.093	0.610
Accuracy Majority Voting	0.543 ± 0.091	0.575 ± 0.093	0.615
Balanced Accuracy Majority Voting	0.544 ± 0.084	0.610 ± 0.071	0.643
Precision Majority Voting (Macro)	0.557 ± 0.093	0.686 ± 0.044	0.773
F1-score Majority Voting (Macro)	0.531 ± 0.082	0.536 ± 0.128	0.575
F1-score Majority Voting (Weighted)	0.533 ± 0.084	0.521 ± 0.142	0.565

Table 25: Test-level comparison of the three main network models, 2D-CNN, 2D-CNN + BiLSTM, and Wav2Vec 2.0, for 2 class classification (nfvPPA and svPPA)

Test set Task 8 nfvPPA vs lvPPA	2D-CNN (12s, overlap 6s, normalization pre)	2D-CNN + biLSTM (6s, overlap 3s, normalization post)	Wav2vec2.0 (12s, overlap 6s, normalization post)
Accuracy	0.706 ± 0.080	0.747 ± 0.019	0.669
Balanced Accuracy	0.698 ± 0.068	0.739 ± 0.013	0.666
Precision (Macro)	0.708 ± 0.077	0.745 ± 0.018	0.670
F1-score (Macro)	0.697 ± 0.074	0.740 ± 0.014	0.666
F1-score (Weighted)	0.704 ± 0.077	0.748 ± 0.015	0.667
Accuracy Majority Voting	0.741 ± 0.094	0.759 ± 0.026	0.636
Balanced Accuracy Majority Voting	0.717 ± 0.074	0.728 ± 0.027	0.660
Precision Majority Voting (Macro)	0.742 ± 0.087	0.743 ± 0.034	0.650
F1-score Majority Voting (Macro)	0.717 ± 0.086	0.728 ± 0.027	0.633
F1-score Majority Voting (Weighted)	0.736 ± 0.090	0.756 ± 0.022	0.642

Table 26: Test-level comparison of the three main network models, 2D-CNN, 2D-CNN + BiLSTM, and Wav2Vec 2.0, for 2 class classification (nfvPPA and lvPPA).

Test set Task 9 lvPPA vs svPPA	2D-CNN (12s, overlap 6s, normalization pre)	2D-CNN + biLSTM (6s, overlap 3s, normalization post)	Wav2vec2.0 (12s, overlap 6s, normalization post)
Accuracy	0.494 ± 0.041	0.575 ± 0.018	0.556
Balanced Accuracy	0.490 ± 0.007	0.516 ± 0.013	0.441
Precision (Macro)	0.489 ± 0.007	0.459 ± 0.113	0.414
F1-score (Macro)	0.455 ± 0.037	0.450 ± 0.058	0.419
F1-score (Weighted)	0.463 ± 0.062	0.491 ± 0.034	0.517
Accuracy Majority Voting	0.553 ± 0.014	0.575 ± 0.018	0.600
Balanced Accuracy Majority Voting	0.525 ± 0.007	0.516 ± 0.013	0.500
Precision Majority Voting (Macro)	0.535 ± 0.006	0.459 ± 0.113	0.300
F1-score Majority Voting (Macro)	0.500 ± 0.029	0.450 ± 0.058	0.375
F1-score Majority Voting (Weighted)	0.520 ± 0.022	0.491 ± 0.034	0.450

Table 27: Test-level comparison of the three main network models, 2D-CNN, 2D-CNN + BiLSTM, and Wav2Vec 2.0, for 2 class classification (lvPPA and svPPA).

6.3 T1 MRI-Based Model Results

The 3D-CNN model trained on patients' 3D T1-weighted images is presented, with a detailed evaluation of both its performance metrics and explainability.

6.3.1 3D MRI Model Performance

The following section reports the performance of the 3D-CNN network on the MRI dataset. Evaluations were conducted for all the tasks shown in Table 5.

6.3.1.1 Multiclass Classification: HC, nfvPPA, svPPA, lvPPA

The following Table 28 reports the performance of the 3D-CNN network applied to the subjects' MRI scans. The network achieved a balanced accuracy of 0.694, a Macro F1-score of 0.729, and a Weighted F1-score of 0.727 on Task 1, confirming its discriminative ability among the considered classes.

Test set Task 1: HC, nfvPPA, svPPA and lvPPA	3D-CNN
Accuracy	0.743
Balanced Accuracy	0.694
Precision (Macro)	0.816
F1-score (Macro)	0.729
F1-score (Weighted)	0.727

Table 28: 3D-CNN Performance on the Test Set for 4 Classes (HC, nfvPPA, svPPA, and lvPPA)

6.3.1.2 Multiclass Classification: All PPA Variant

The results of Task 2, applied to the 3D-CNN network, are show in Table 29.

Test set Task 2: nfvPPA, svPPA, lvPPA	3D-CNN
Accuracy	0.889
Balanced Accuracy	0.875
Precision (Macro)	0.869
F1-score (Macro)	0.869
F1-score (Weighted)	0.889

Table 29: 3D-CNN Performance on the Test Set for the 3 PPA variants (nfvPPA, svPPA, and lvPPA).

6.3.1.3 Binary Classification of HC vs PPA: Overall and Variant-specific Analysis

The Table 30 reports the results of the 3D-CNN network applied in a binary approach, specifically for Tasks 3, 4, 5, and 6.

Test set	3D-CNN Task 3: HC vs PPA	3D-CNN Task 4: HC vs nfvPPA	3D-CNN Task 5: HC vs svPPA	3D-CNN Task 6: HC vs lvPPA
Accuracy	0.796 ± 0.030	0.808 ± 0.047	0.948 ± 0.001	0.575 ± 0.018
Balanced Accuracy	0.799 ± 0.035	0.725 ± 0.059	0.909 ± 0.000	0.516 ± 0.013
Precision (Macro)	0.812 ± 0.038	0.813 ± 0.077	0.966 ± 0.001	0.459 ± 0.113
F1-score (Macro)	0.794 ± 0.031	0.746 ± 0.066	0.932 ± 0.000	0.450 ± 0.058
F1-score (Weighted)	0.794 ± 0.030	0.793 ± 0.052	0.946 ± 0.001	0.491 ± 0.034

Table 30: 3D-CNN Performance on the Test Set for Binary Classification of HC vs PPA, Including Overall and Variant-specific Comparisons: HC vs nfvPPA, HC vs svPPA, HC vs lvPPA.

6.3.1.4 Binary Classification Between PPA Variants

The Table 31 reports the results of the 3D-CNN network applied in a binary approach, where two PPA variants are compared at a time, specifically for Tasks 7, 8, and 9.

Test set	3D-CNN Task 7: svPPA vs nfvPPA	3D-CNN Task 8: lvPPA vs nfvPPA	3D-CNN Task 9: lvPPA vs svPPA
Accuracy	0.944 ± 0.019	0.575 ± 0.018	0.729 ± 0.103
Balanced Accuracy	0.942 ± 0.023	0.516 ± 0.013	0.707 ± 0.126
Precision (Macro)	0.951 ± 0.013	0.459 ± 0.113	0.708 ± 0.127
F1-score (Macro)	0.943 ± 0.020	0.450 ± 0.058	0.702 ± 0.125
F1-score (Weighted)	0.943 ± 0.020	0.491 ± 0.034	0.726 ± 0.107

Table 31: 3D-CNN Performance on the Test Set for binary classification between PPA Variants, including nfvPPA vs svPPA, lvPPA vs svPPA, and nfvPPA vs lvPPA.

6.4 Model explainability

This chapter presents examples of interpretability techniques applied to models. Specifically, examples of Grad-CAM applied to 2D CNNs are shown both in the multi-class setting (Figure 81) and in the binary comparison between HC and PPA (Figure 82). Next, examples of Grad-CAM applied to 3D CNNs are illustrated in the binary

comparison between HC and lvPPA (Figure 83). Finally, examples of Integrated Gradients applied to Wav2Vec are reported in the comparison between HC and PPA (Figure 84).

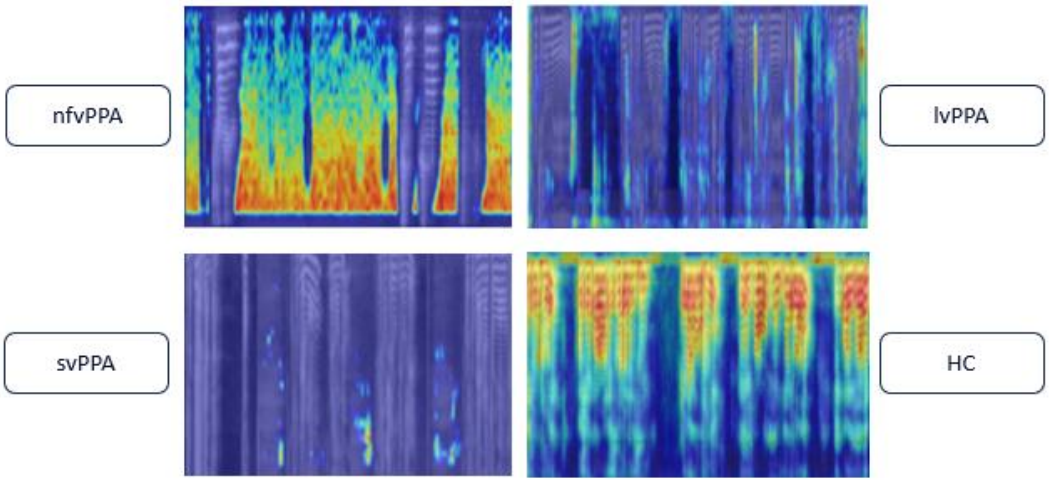

Figure 81: Grad-CAM applied to the 2D-CNN model on spectrograms from nfvPPA, lvPPA, svPPA, and HC subjects, highlighting the regions most relevant for classification.

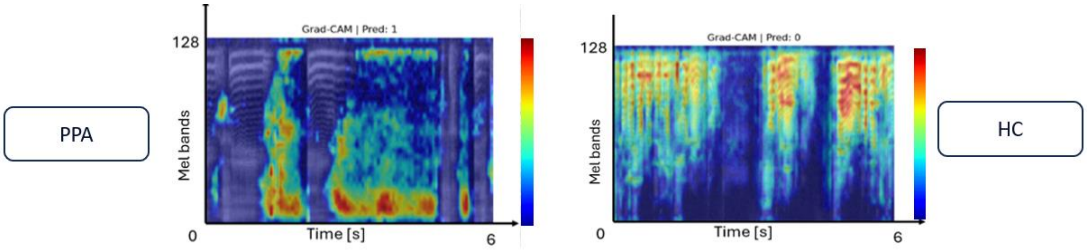

Figure 82: Grad-CAM applied to the 2D-CNN model on spectrograms from PPA and HC subjects, highlighting the regions most relevant for classification.

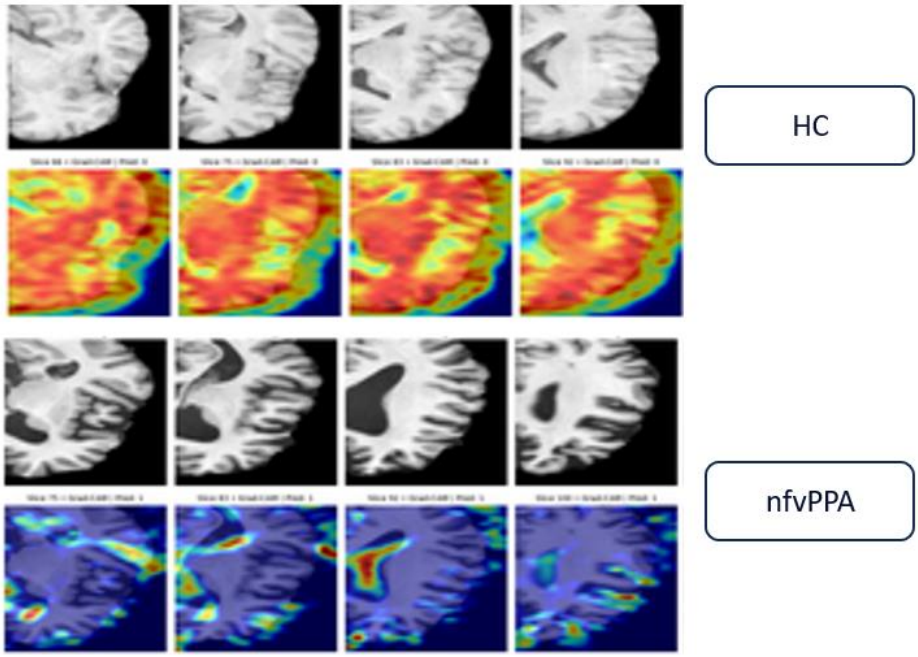

Figure 83: Grad-CAM applied to the 3D-CNN model o on T1-weighted MRI images from nfvPPA and HC subjects, highlighting the regions most important for classification.

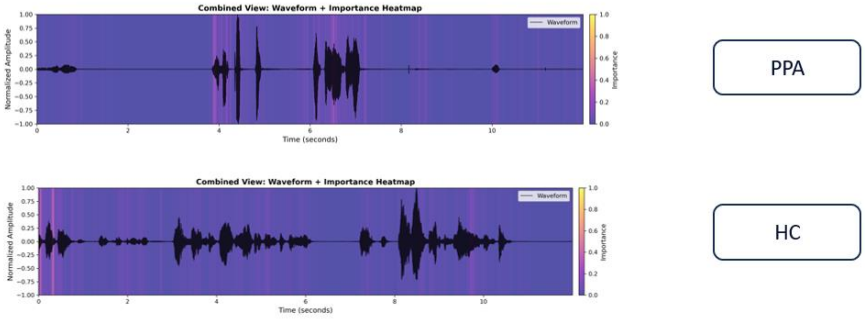

Figure 84: Integrated Gradients applied to the Wav2Vec2.0 model on waveforms from PPA and HC subjects, highlighting the most important regions for classification.

6.5 Multimodal Integration Results

Table 32 and Table 33 show the results of Task 10, corresponding to the fusion of the networks. Each table presents the test set values for both networks before fusion, followed by the results obtained after fusion.

Table 32 reports the performance of the network fusion between CNN2D + BiLSTM and CNN3D. The network was trained using the combined predictions on the validation set, and the optimization process identified an optimal weight of $w = 0.21$, assigning 0.21 to the predictions of the audio network and 0.79 to those of the image-based network. The final balanced accuracy achieved by the fusion is 0.830 on the test set.

The same approach is also applied in Table 33, where the fusion of the Wav2Vec2.0 and CNN3D networks is shown. In this case, the optimal weight selected was $w = 0.32$ for the audio network and 0.68 for the image network. The fusion achieved a balanced accuracy of 0.938 on the test set.

Test set Task 10	2D-CNN + BiLSTM	3D-CNN	Fusion
Accuracy	0.611	0.722	0.778
Balanced Accuracy	0.622	0.795	0.830
Precision (Macro)	0.643	0.861	0.875
F1-score (Macro)	0.622	0.795	0.813
F1-score (Weighted)	0.619	0.727	0.788

Table 32: Performance metrics of 2D-CNN + BiLSTM, 3D-CNN, and fused multimodal model on the test set.

Test set Task 10	Wav2vec2.0	3D-CNN	Fusion
Accuracy	0.722	0.722	0.994
Balanced Accuracy	0.720	0.795	0.938
Precision (Macro)	0.708	0.861	0.950
F1-score (Macro)	0.720	0.795	0.937
F1-score (Weighted)	0.715	0.727	0.944

Table 33: Performance metrics of Wav2vec2.0, 3D-CNN, and fused multimodal model on the test set.

7. Discussion and Conclusion

7.1 Discussion

The main objective of this thesis is to develop a deep learning network capable of classifying subjects affected by PPA. The initial approach involved creating deep learning networks adapted for multiclass classification, aiming to distinguish healthy controls from the three PPA variants: non-fluent variant, semantic variant, and logopenic variant; the same networks were then tested in binary classification settings. As highlighted in the literature, for example in the study by Vignesh Subramanian et al. [130], multiclass classification represents a greater challenge compared to binary classification, especially because, as the number of classes increases, the number of positive examples available for each class decreases, making the learning of over-parameterized models more complex.

To address this challenge, this thesis proposes a multimodal approach, combining information obtained from patients' speech and three-dimensional brain magnetic resonance images. The idea was to build neural networks specialized for each type of data and then fuse them to achieve improved performance and a comprehensive integration of multimodal information.

For speech processing, the thesis took as a reference the study by Daniel Escobar-Grisales et al. [97], which proposes several models for the classification of Parkinson's disease based on vocal signals. In line with this approach, three types of neural networks were developed: a 2D CNN network, a 2D CNN network with the addition of a BiLSTM layer, and a network based on a pre-trained speech model, Wav2vec.

For the 2D CNN networks, the audio was transformed into spectrograms. Among the representations tested, log-Mel spectrograms showed the best performance compared to Mel spectrograms and MFCCs. This result is consistent with the literature: in the study by Wei Chen et al. [131], the superiority of log-Mel spectrograms over Mel spectrograms is highlighted; similarly, in the study by Suhas BN et al. [101], log-Mel spectrograms are compared with MFCCs, once again demonstrating the superiority of the log-Mel spectrogram representation.
The preprocessing of the audio was adapted to the requirements of each architecture. For the 2D CNN, normalization was applied to the entire audio, followed by segmentation into 12-second segments, thus providing the network with longer temporal sequences. This approach allows the 2D CNN to exploit a larger amount of information, highlighting its ability to learn relevant patterns especially in extended

signal windows. In contrast, for the 2D CNN + BiLSTM network, the best approach consists of segmenting the audio into shorter segments of 6 seconds, with normalization subsequently applied to each individual fragment. This difference between the 2D CNN and the 2D CNN + BiLSTM network partly reflects what was observed by Bai et al.[132], according to which convolutional networks perform better with long sequences to capture long-term dependencies, whereas recurrent networks, including LSTMs, show instability on extended sequences and prefer shorter segments for effective memory. Therefore, the 2D CNN benefits from long sequences, while the addition of the BiLSTM optimizes performance on shorter sequences.

Subsequently, after determining the most suitable audio preprocessing for the two networks, attempts were made to penalize the most severe errors and to apply data augmentation for both the CNN2D network and the CNN2D+Bilstm network. However, these strategies did not lead to improved performance, supporting the continuation of testing with a more standard and simpler approach, without employing these techniques.

The best performance in terms of multi-class classification was achieved by the 2D CNN with the addition of the BiLSTM layer, reaching a balanced accuracy of 0.624, a weighted F1-score of 0.645, and a Macro F1-score of 0.610. This performance was obtained using a final majority voting approach, which aggregates the classifications of each audio segment and assigns a final label for each subject. Among binary comparisons, the performances of the evaluated models were generally comparable, with the simpler 2D CNN network sometimes achieving slightly better results.This behavior is consistent with the literature, as increasing model complexity in simple tasks can lead to a decrease in performance. This concept is also discussed by Oyedare et al. [133], where the study emphasizes how simpler models on simpler tasks achieve higher performance and greater generalization capability. Similarly, in the work of Abdullah et al. [134] it is highlighted how a CNN can outperform an LSTM when temporal dependencies are not complex. Consequently, in the binary context, where complex temporal information is not required, the 2D CNN can sometimes outperform the CNN2D + biLSTM model.

Subsequently, after the creation of the two models, it was decided to proceed with a final model for the audio network, based on the use of a Wav2vec network. The best preprocessing once again involved relatively long audio segments of 12 s, with normalization applied to each fragment.
This approach provided the best performance, achieving a balanced accuracy of 0.745 a Weighted F1-score of 0.772, and a Macro F1-score of 0.745, again after applying a majority voting step. The superiority of Wav2vec is consistent with the findings reported in the initial study by Daniel Escobar-Grisales et al.[97].

From the results obtained with the three networks, a common observation emerges: in binary comparisons, distinguishing between a PPA variant and HC is easier for all three models, resulting in higher performance and highlighting their ability to separate healthy speech from pathological speech. Conversely, distinguishing between two PPA variants is more challenging, leading to lower performance. An important finding emerges in the classification between nfvPPA and lvPPA, a distinction that is often difficult even for clinicians. Despite the complexity of this task, the networks achieve good performance, with the highest results obtained by the 2D-CNN + BiLSTM model: a balanced accuracy of 0.72 ± 0.027 and a weighted F1-score of 0.756 ± 0.022.

Regarding the image-based network, taking as a reference the study by Basaia et al. [92], an optimal configuration was developed capable of processing 3D T1 images as input, applying preprocessing that includes normalization in MNI space and patch creation. This configuration, which also included a data augmentation step, achieved multi-class performance of 0.694 in balanced accuracy, 0.727 in Weighted F1-score, and 0.729 in Macro F1-score. In this network, the binary comparisons appear equally stable, suggesting that the model does not show differences in error rates between HC and a PPA variant, nor between two PPA variants, unlike the audio-based network, where such differences are more pronounced. A notable detail is that the image-based network performs well in discriminating between svPPA and nfvPPA (balanced accuracy = 0.942) and between svPPA and lvPPA (balanced accuracy = 0.707). However, performance is lower when distinguishing between nfvPPA and lvPPA, reaching only around 0.516 in balanced accuracy. Interestingly, this pattern is opposite to that observed in the audio-based networks, which performed well in the latter comparison but poorly in the first two. Such complementary behavior suggests a promising outcome for multimodal integration.

The explainability technique was subsequently applied to the different networks to effectively evaluate their focus. Grad-CAM was used for both the 2D and 3D CNNs, while Integrated Gradients was applied to wav2vec2. Applying Grad-CAM to the 2D CNN revealed interesting patterns when comparing HC and PPA subjects. In HC subjects, the network primarily focuses on the high-frequency components of the spectrogram, corresponding to a perceptually high pitch, which is consistent with the speech characteristics of healthy individuals, typically exhibiting higher pitch and greater energy. In PPA subjects, the network's attention is mainly directed toward low-frequency components and pauses, reflecting the characteristics of the pathology, as their speech tends to be slower, fragmented, and monotonous. When examining the individual PPA variants, it was observed that in the logopenic variant the network focuses on pauses across all frequencies, corresponding to word-finding difficulties, brief hesitations, or micro-pauses throughout speech; in the non-fluent variant attention is mainly directed toward low frequencies, highlighting longer and more marked pauses, typical of fragmented speech; while in the semantic variant the focus

appears only in brief, punctuated moments, corresponding to hesitations, sighs, or moments of uncertainty, reflecting semantic search. These patterns suggest that the network automatically captures the distinctive features of each PPA variant. At the 3D CNN level, the explainability model showed that the network primarily concentrates on brain regions most affected by atrophy, and specifically for the logopenic variant attention is directed toward the ventricles and general brain atrophy. Finally, for wav2vec2, interpretability is more challenging as the focus remains exclusively on speech, recognizing subjects solely from their speech patterns without leveraging information from pauses.

Integration between networks was subsequently performed by combining the CNN2D+BiLSTM with the CNN3D, and the Wav2vec network with the CNN3D. The fusion strategy adopted was late fusion, chosen both for the simplicity of the approach and for the low associated computational cost. The fusion of CNN2D+BiLSTM and CNN3D produced a balanced accuracy of 0.830, a Weighted F1-score of 0.788, and a Macro F1-score of 0.813, while the combination of Wav2vec and CNN3D achieved a balanced accuracy of 0.938, a Weighted F1-score of 0.944, and a Macro F1-score of 0.937.

These results demonstrate how a late fusion approach is capable of improving performance, in line with the findings reported in the study by Ioannis Galanakis et al.[108]

Two different networks were chosen for the fusion phase, namely both the CNN2D+BiLSTM and Wav2vec. Although the CNN2D+BiLSTM has slightly lower performance, it presents significantly shorter training times, approximately 6–8 hours per network. In contrast, the use of Wav2vec, while providing superior performance, requires a very long training time, approximately 36 hours per network, with a consequently high computational and energy cost.

7.2 Limitations

Several limitations must be considered when interpreting the results of this case study.

The first issue concerns the presence of a very small dataset, both for audio and imaging networks. Deep learning models generally require large and balanced datasets in order to generalize effectively. A small and imbalanced dataset can lead to several undesirable effects, including instability during training, difficulty in hyperparameter estimation, and significant variability across different splits.
To mitigate this limitation, a class-balanced split was applied during the creation of the training and validation sets, and weights proportional to the number of samples per class were assigned during training. More frequent classes were assigned lower

weights, while underrepresented classes received higher weights, so that the model would pay more attention to minority classes without neglecting the majority ones.

Despite these measures, the small dataset size amplified the variability of performance: even a single misclassified sample could have a substantial impact on accuracy. To address this, a hybrid nested cross-validation approach was adopted to better assess the model's generalization ability.

However, this approach showed limitations in the multiclass scenario, both for audio and imaging networks. Each class contained few examples, and splitting into folds could result in validation sets that were not fully representative, compromising hyperparameter estimation and training stability.

Even in the binary scenario, particularly with audio data, it was observed that depending on the split, the model could fail to train properly, sometimes leading to phenomena such as automatic learning rate increases. This suggests that the split should also consider subject-specific characteristics. Attempts were made to balance the input data based on audio duration, number of pauses, and number of words, but these strategies did not yield significant improvements. The split therefore remains an open problem, which could likely be resolved by increasing the number of subjects.

7.3 Conclusion

In conclusion, this thesis demonstrates that deep learning can be successfully applied to the diagnosis of PPA. The use of networks based on patient speech enables the classification of the disease and its variants, while MRI image analysis allows accurate classification. The integration of the two information modalities through late fusion provides a further increase in performance, highlighting the potential of a multimodal approach for the automated diagnosis of PPA.

Appendix

The following plots show the loss trends over the epochs for the four multiclass classifiers: 2D CNN, 2D CNN with BiLSTM, Wav2Vec2.0 and 3D CNN.

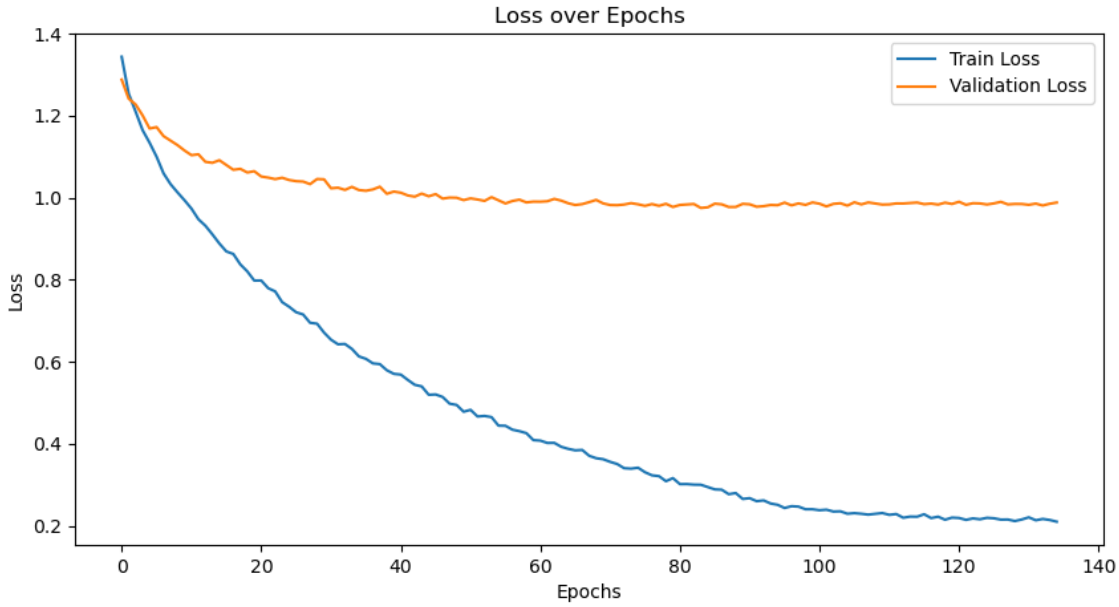

Figure 85: 2D CNN (250 epochs with early stop): Trend of training and validation loss over the epochs.

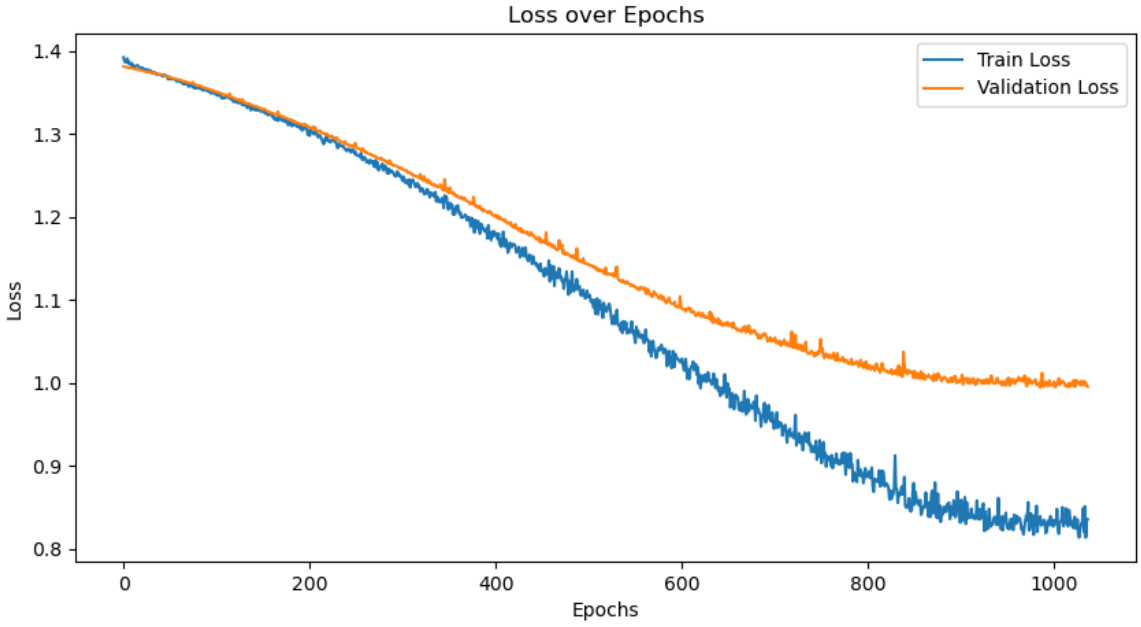

Figure 86: 2D CNN + BiLSTM (1500 epochs with early stop): Trend of training and validation loss over the epochs.

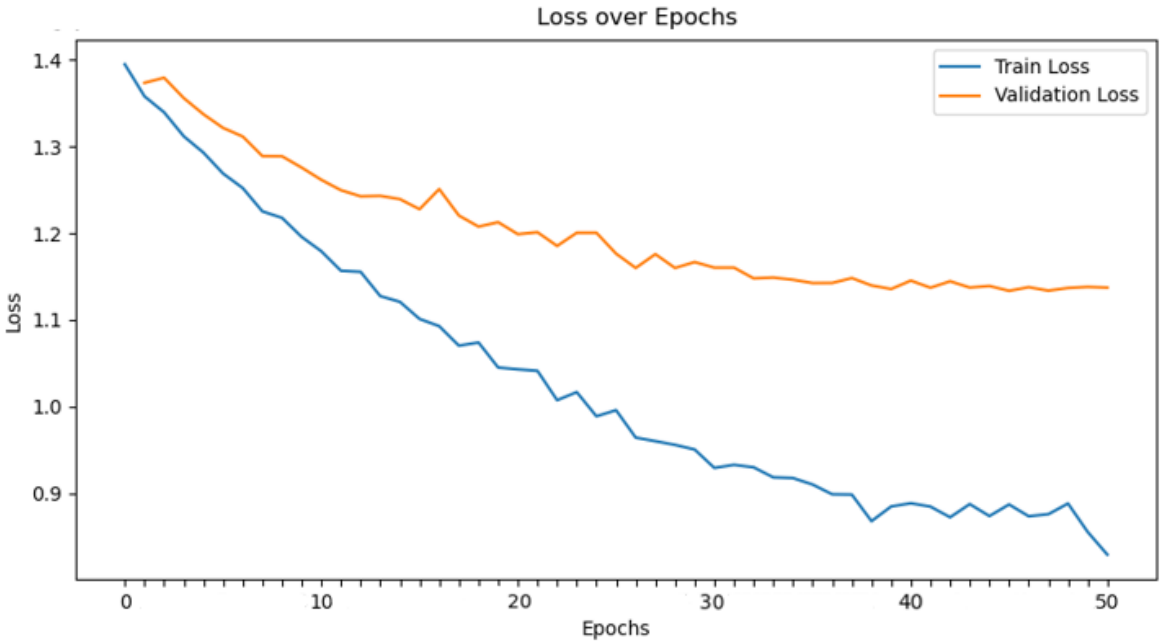

Figure 87: Wav2vec2.0 (50 epochs): Trend of training and validation loss over the epochs.

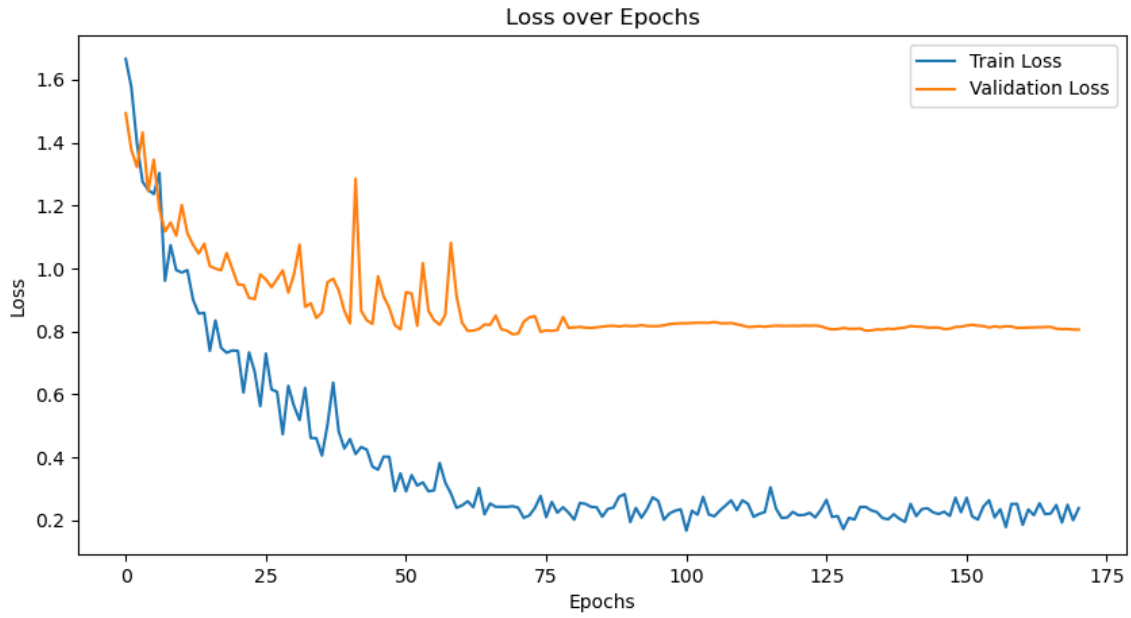

Figure 88: 3D-CNN (500 epochs): Trend of training and validation loss over the epochs.

Bibliography

1. Gorno-Tempini, M. L. *et al.* Classification of primary progressive aphasia and its variants. *Neurology* **76**, 1006–1014 (2011).
2. Léger, G. C. & Johnson, N. A review on primary progressive aphasia. *Neuropsychiatr. Dis. Treat.* **3**, 745–752 (2007).
3. Khan, I. & De Jesus, O. Frontotemporal Lobe Dementia. in *StatPearls* (StatPearls Publishing, Treasure Island (FL), 2025).
4. Olney, N. T., Spina, S. & Miller, B. L. Frontotemporal Dementia. *Neurol. Clin.* **35**, 339–374 (2017).
5. Leyton, C. & Ballard, K. Primary progressive aphasia: conceptual evolution and challenges. *Neurosci. Neuroeconomics* **2016**, 9 (2016).
6. Conca, F., Esposito, V., Giusto, G., Cappa, S. F. & Catricalà, E. Characterization of the logopenic variant of Primary Progressive Aphasia: A systematic review and meta-analysis. *Ageing Res. Rev.* **82**, 101760 (2022).
7. Bekkhus-Wetterberg, P. *et al.* Primary progressive aphasia. *Tidsskr. Den Nor. Legeforening* https://doi.org/10.4045/tidsskr.22.0100 (2022) doi:10.4045/tidsskr.22.0100.
8. Ortiz, G. G. *et al.* Primary Progressive Aphasias: Diagnosis and Treatment. *Brain Sci.* **15**, 245 (2025).
9. Turcano, P. *et al.* Incidence of Primary Progressive Apraxia of Speech and Primary Progressive Aphasia in Olmsted County, MN, 2011–2022. *Neurology* **103**, e209693 (2024).
10. Ulugut, H. *et al.* The natural history of primary progressive aphasia: beyond aphasia. *J. Neurol.* **269**, 1375–1385 (2022).

11. Maldonado, K. A. & Alsayouri, K. Physiology, Brain. in *StatPearls* (StatPearls Publishing, Treasure Island (FL), 2025).
12. Jawabri, K. H. & Sharma, S. Physiology, Cerebral Cortex Functions. in *StatPearls* (StatPearls Publishing, Treasure Island (FL), 2025).
13. Alzaidi, S. S. Computational Models of Cerebral Hemodynamics.
14. Ramoo, D. 4.4 Language in the Brain. https://opentextbc.ca/psyclanguage/chapter/language-in-the-brain/ (2021).
15. Kiymaz, T., Khan Suheb, M. Z., Lui, F. & De Jesus, O. Primary Progressive Aphasia. in *StatPearls* (StatPearls Publishing, Treasure Island (FL), 2025).
16. (PDF) Primary Progressive Aphasia and the Left Hemisphere Language Network. https://www.researchgate.net/publication/313482317_Primary_Progressive_Aphasia_and_the_Left_Hemisphere_Language_Network.
17. Routier, A. *et al.* Structural, Microstructural, and Metabolic Alterations in Primary Progressive Aphasia Variants. *Front. Neurol.* **9**, 766 (2018).
18. Eh, de F., Af, B. & Tm, D. Basic concepts of MR imaging, diffusion MR imaging, and diffusion tensor imaging. *Magn. Reson. Imaging Clin. N. Am.* **19**, (2011).
19. Forshult, S. E. *Magnetic Resonance Imaging: MRI: An Overview*. (Faculty of Technology and Science, Physics, Karlstad University, Karlstad, 2007).
20. Evers, H., Hawighorst, H., van Kaick, G., Knapstein, P. G. & Meinzer, H. P. [Integration of functional and morphologic MRI data for preoperative 3D visualization of tumors. Example: cervical carcinoma]. *Radiol.* **38**, 841–847 (1998).

21. Grover, V. P. B. *et al.* Magnetic Resonance Imaging: Principles and Techniques: Lessons for Clinicians. *J. Clin. Exp. Hepatol.* **5**, 246–255 (2015).

22. 5.1: Nuclear Spin and Magnetic Field. *Chemistry LibreTexts* https://chem.libretexts.org/Courses/Western_Washington_University/Biophysical_Chemistry_(Smirnov_and_McCarty)/05%3A_Nuclear_Magnetic_Resonance_(NMR)_Spectroscopy_-_Introduction/5.01%3A_Nuclear_Spin_and_Magnetic_Field (2022).

23. Khashami, F. A mini review of NMR and MRI. Preprint at https://doi.org/10.48550/arXiv.2401.01389 (2024).

24. van Geuns, R.-J. M. *et al.* Basic principles of magnetic resonance imaging. *Prog. Cardiovasc. Dis.* **42**, 149–156 (1999).

25. An, H. & Lin, W. Spin Density, T_1 , T_2 , T_2 * Relaxation and Bloch Equations. *Curr. Protoc. Magn. Reson. Imaging* **00**, (2001).

26. Jackson, E. F., Ginsberg, L. E., Schomer, D. F. & Leeds, N. E. A review of MRI pulse sequences and techniques in neuroimaging. *Surg. Neurol.* **47**, 185–199 (1997).

27. A mini review of NMR and MRI. https://arxiv.org/html/2401.01389v1.

28. Plewes, D. B. & Kucharczyk, W. Physics of MRI: A primer. *J. Magn. Reson. Imaging* **35**, 1038–1054 (2012).

29. In vivo NMR Imaging: Methods and Protocols | SpringerLink. https://link.springer.com/book/10.1007/978-1-61779-219-9.

30. Gallagher, T. A., Nemeth, A. J. & Hacein-Bey, L. An Introduction to the Fourier Transform: Relationship to MRI. *Am. J. Roentgenol.* **190**, 1396–1405 (2008).

31. Al-Tamimi, M. Survey Based Study: Classification of Patients with Alzheimer's Disease . *Iraqi J. Sci.* **61**, 3104–3126 (2020).

32. Hirsch, G. V., Bauer, C. M. & Merabet, L. B. Using structural and functional brain imaging to uncover how the brain adapts to blindness. *Ann. Neurosci. Psychol.* **2**, 5 (2015).

33. Roytman, M., Chiang, G. C., Gordon, M. L. & Franceschi, A. M. Multimodality Imaging in Primary Progressive Aphasia. *Am. J. Neuroradiol.* **43**, 1230–1243 (2022).

34. Mandelli, M. L. *et al.* Healthy brain connectivity predicts atrophy progression in non-fluent variant of primary progressive aphasia. *Brain J. Neurol.* **139**, 2778–2791 (2016).

35. Mandelli, M. L. *et al.* Two insular regions are differentially involved in behavioral variant FTD and nonfluent/agrammatic variant PPA. *Cortex J. Devoted Study Nerv. Syst. Behav.* **74**, 149–157 (2016).

36. Agosta, F. *et al.* Differentiation between Subtypes of Primary Progressive Aphasia by Using Cortical Thickness and Diffusion-Tensor MR Imaging Measures. *Radiology* **276**, 219–227 (2015).

37. Yang, J., Pan, P., Song, W. & Shang, H.-F. Quantitative meta-analysis of gray matter abnormalities in semantic dementia. *J. Alzheimers Dis. JAD* **31**, 827–833 (2012).

38. Whitwell, J. L. FTD spectrum: Neuroimaging across the FTD spectrum. *Prog. Mol. Biol. Transl. Sci.* **165**, 187–223 (2019).

39. Peet, B. T., Spina, S., Mundada, N. & La Joie, R. Neuroimaging in Frontotemporal Dementia: Heterogeneity and Relationships with Underlying Neuropathology. *Neurother. J. Am. Soc. Exp. Neurother.* **18**, 728–752 (2021).

40. Madhavan, A. *et al.* FDG PET and MRI in logopenic primary progressive aphasia versus dementia of the Alzheimer's type. *PloS One* **8**, e62471 (2013).

41. Tozzi, A. E. Introduzione all'intelligenza artificiale in medicina per il personale sanitario.

42. Stecher, M. T. La storia dell'intelligenza artificiale, da Turing ad oggi. *CyberLaws* https://www.cyberlaws.it/2018/la-storia-dellintelligenza-artificiale-da-turing-ad-oggi/ (2018).

43. Choi, R. Y., Coyner, A. S., Kalpathy-Cramer, J., Chiang, M. F. & Campbell, J. P. Introduction to Machine Learning, Neural Networks, and Deep Learning. *Transl. Vis. Sci. Technol.* **9**, 14.

44. Kufel, J. *et al.* What Is Machine Learning, Artificial Neural Networks and Deep Learning?—Examples of Practical Applications in Medicine. *Diagnostics* **13**, 2582 (2023).

45. Ghosh, M. & Arunachalam, T. Introduction to Artificial Intelligence. in 23–44 (2021). doi:10.1007/978-981-16-0415-7_2.

46. Ashmore, R., Calinescu, R. & Paterson, C. Assuring the Machine Learning Lifecycle: Desiderata, Methods, and Challenges. Preprint at https://doi.org/10.48550/arXiv.1905.04223 (2019).

47. Categorical Data Encoding Techniques in Machine Learning. *GeeksforGeeks* https://www.geeksforgeeks.org/machine-learning/categorical-data-encoding-techniques-in-machine-learning/ (18:01:53+00:00).

48. Feature Extraction in Machine Learning: A Complete Guide. https://www.datacamp.com/tutorial/feature-extraction-machine-learning.

49. Sarker, I. H. Machine Learning: Algorithms, Real-World Applications and Research Directions. *Sn Comput. Sci.* **2**, 160 (2021).

50. Liu, X., Qi, H., Jia, S., Guo, Y. & Liu, Y. Recent Advances in Optimization Methods for Machine Learning: A Systematic Review. *Mathematics* **13**, 2210 (2025).

51. Burkov, A. “All models are wrong, but some are useful.” — George Box.

52. Simeone, O. A Very Brief Introduction to Machine Learning With Applications to Communication Systems. Preprint at https://doi.org/10.48550/arXiv.1808.02342 (2018).

53. Oti, E., Olusola, M., Eze, F. & Enogwe, S. Comprehensive Review of K-Means Clustering Algorithms. *Int. J. Adv. Sci. Res. Eng.* **07**, 64–69 (2021).

54. Verbraeken, J. *et al.* A Survey on Distributed Machine Learning. *ACM Comput. Surv.* **53**, 1–33 (2021).

55. AI, T. A. What is Generalization in Machine Learning? *Applied AI Blog* https://www.appliedaicourse.com/blog/generalization-in-machine-learning/ (2025).

56. Murphy, K. P. *Machine Learning: A Probabilistic Perspective*. (MIT Press, Cambridge, Mass., 2013).

57. Optimization Algorithms in Machine Learning. *GeeksforGeeks* https://www.geeksforgeeks.org/machine-learning/optimization-algorithms-in-machine-learning/ (15:18:03+00:00).

58. Xu, Y. & Goodacre, R. On Splitting Training and Validation Set: A Comparative Study of Cross-Validation, Bootstrap and Systematic Sampling for Estimating the Generalization Performance of Supervised Learning. *J. Anal. Test.* **2**, 249–262 (2018).

59. Adin, A. *et al.* Automatic cross-validation in structured models: Is it time to leave out leave-one-out? *Spat. Stat.* **62**, 100843 (2024).

60. Gholamiangonabadi, D., Kiselov, N. & Grolinger, K. Deep Neural Networks for Human Activity Recognition With Wearable Sensors: Leave-One-Subject-Out Cross-Validation for Model Selection. *IEEE Access* **8**, 133982–133994 (2020).
61. Ellis, C. M. Machine Learning.
62. zhong, Y., He, J. & Chalise, P. Nested and Repeated Cross Validation for Classification Model With High-Dimensional Data. *Rev. Colomb. Estad.* **43**, 103–125 (2020).
63. Badillo, S. *et al.* An Introduction to Machine Learning. *Clin. Pharmacol. Ther.* **107**, 871–885 (2020).
64. Swaminathan, S. & Tantri, B. R. Confusion Matrix-Based Performance Evaluation Metrics. *Afr. J. Biomed. Res.* **27**, 4023–4031 (2024).
65. Sarker, I. H. Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions. *SN Comput. Sci.* **2**, 420 (2021).
66. Qamar, R. & Zardari, B. Artificial Neural Networks: An Overview. *Mesopotamian J. Comput. Sci.* **2023**, 130–139 (2023).
67. Dubey, S. R., Singh, S. K. & Chaudhuri, B. B. Activation Functions in Deep Learning: A Comprehensive Survey and Benchmark. Preprint at https://doi.org/10.48550/arXiv.2109.14545 (2022).
68. Thakur, D., Saini, J. & Srinivasan, S. DeepThink IoT: The Strength of Deep Learning in Internet of Things. *Artif. Intell. Rev.* **56**, (2023).
69. Han, S.-H., Kim, K. W., Kim, S. & Youn, Y. C. Artificial Neural Network: Understanding the Basic Concepts without Mathematics. *Dement. Neurocognitive Disord.* **17**, 83–89 (2018).

70. Razavi, S. Deep learning, explained: Fundamentals, explainability, and bridgeability to process-based modelling. *Environ. Model. Softw.* **144**, 105159 (2021).

71. Chen, B., Deng, W. & Du, J. *Noisy Softmax: Improving the Generalization Ability of DCNN via Postponing the Early Softmax Saturation*. (2017). doi:10.1109/CVPR.2017.428.

72. Mienye, I. D., Swart, T. G. & Obaido, G. Recurrent Neural Networks: A Comprehensive Review of Architectures, Variants, and Applications. *Information* **15**, 517 (2024).

73. Piccioni Costa, L. *et al.* Multilayer Perceptron. in 105 (2023).

74. Terven, J., Cordova-Esparza, D.-M., Ramirez-Pedraza, A. & Chávez Urbiola, E. *Loss Functions and Metrics in Deep Learning. A Review*. (2023). doi:10.48550/arXiv.2307.02694.

75. Ruder, S. An overview of gradient descent optimization algorithms. Preprint at https://doi.org/10.48550/arXiv.1609.04747 (2017).

76. Altınel, D. Development of Deep Learning Optimizers: Approaches, Concepts, and Update Rules. Preprint at https://doi.org/10.48550/arXiv.2509.18396 (2025).

77. Chugani, V. A Gentle Introduction to Learning Rate Schedulers. *MachineLearningMastery.com* https://machinelearningmastery.com/a-gentle-introduction-to-learning-rate-schedulers/ (2025).

78. Ghezzi, S. Inizializzazione dei pesi. *Deep Learning Italia* https://deeplearningitalia.com/inizializzazione-dei-pesi/ (2023).

79. Ioffe, S. & Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.

80. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting.

81. Vakalopoulou, M., Christodoulidis, S., Burgos, N., Colliot, O. & Lepetit, V. *Deep Learning: Basics and Convolutional Neural Networks (CNN)*. (2023).

82. O'Shea, K. & Nash, R. An Introduction to Convolutional Neural Networks. Preprint at https://doi.org/10.48550/arXiv.1511.08458 (2015).

83. Michele, B. TECNICHE RICOSTRUZIONE IMMAGINI TOMOGRAFICHE. *Xrayconsult* https://www.xrayconsult.it/nota-tomo-tecniche.html.

84. Purwono, I. *et al.* Understanding of Convolutional Neural Network (CNN): A Review. *Int. J. Robot. Control Syst.* **2**, 739–748 (2023).

85. (PDF) A improved pooling method for convolutional neural networks. https://www.researchgate.net/publication/377498564_A_improved_pooling_method_for_convolutional_neural_networks.

86. Lin, M., Chen, Q. & Yan, S. Network In Network. Preprint at https://doi.org/10.48550/arXiv.1312.4400 (2014).

87. Schmidt, R. M. Recurrent Neural Networks (RNNs): A gentle Introduction and Overview. Preprint at https://doi.org/10.48550/arXiv.1912.05911 (2019).

88. Vennerød, C. B., Kjærran, A. & Bugge, E. S. Long Short-term Memory RNN. Preprint at https://doi.org/10.48550/arXiv.2105.06756 (2021).

89. Vaswani, A. *et al.* Attention Is All You Need. Preprint at https://doi.org/10.48550/arXiv.1706.03762 (2023).

90. Schneider, S., Baevski, A., Collobert, R. & Auli, M. wav2vec: Unsupervised Pre-training for Speech Recognition. Preprint at https://doi.org/10.48550/arXiv.1904.05862 (2019).

91. Baevski, A., Zhou, H., Mohamed, A. & Auli, M. wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations. Preprint at https://doi.org/10.48550/arXiv.2006.11477 (2020).

92. Basaia, S. *et al.* Automated classification of Alzheimer's disease and mild cognitive impairment using a single MRI and deep neural networks. *NeuroImage Clin.* **21**, 101645 (2019).

93. Castellano, G., Esposito, A., Mirizio, M., Montanaro, G. & Vessio, G. *Detection of Dementia Through 3D Convolutional Neural Networks Based on Amyloid PET*. 6 (2021). doi:10.1109/SSCI50451.2021.9660102.

94. Basaia, S. *et al.* Multi-Center 3D CNN for Parkinson's disease diagnosis and prognosis using clinical and T1-weighted MRI data. *NeuroImage Clin.* **48**, 103859 (2025).

95. Deep Learning for Pathological Speech: A Survey. https://arxiv.org/html/2501.03536v1.

96. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* **521**, 436–444 (2015).

97. Escobar-Grisales, D., Ríos-Urrego, C. D. & Orozco-Arroyave, J. R. Deep Learning and Artificial Intelligence Applied to Model Speech and Language in Parkinson's Disease. *Diagnostics* **13**, 2163 (2023).

98. Kilimci, Z. H., Bayraktar, U. & Kucukmanisa, A. Evaluating raw waveforms with deep learning frameworks for speech emotion recognition. Preprint at https://doi.org/10.48550/arXiv.2307.02820 (2023).

99. Lin, K. & Washington, P. Y. Multimodal deep learning for dementia classification using text and audio. *Sci. Rep.* **14**, 13887 (2024).

100. Ahn, K. *et al.* Deep Learning of Speech Data for Early Detection of Alzheimer's Disease in the Elderly. *Bioengineering* **10**, 1093 (2023).

101. Suhas, B. *et al.* Speech task based automatic classification of ALS and Parkinson's Disease and their severity using log Mel spectrograms. in *2020 International Conference on Signal Processing and Communications (SPCOM)* 1–5 (2020). doi:10.1109/SPCOM50965.2020.9179503.

102. Fraser, K., Rudzicz, F., Graham, N. & Rochon, E. Automatic speech recognition in the diagnosis of primary progressive aphasia.

103. Themistocleous, C. *et al.* Automatic Subtyping of Individuals with Primary Progressive Aphasia. *J. Alzheimers Dis.* **79**, 1185–1194.

104. Jiao, T., Guo, C., Feng, X., Chen, Y. & Song, J. A Comprehensive Survey on Deep Learning Multi-Modal Fusion: Methods, Technologies and Applications. *Comput. Mater. Contin.* **80**, 1–35 (2024).

105. Multimodal Alignment and Fusion: A Survey. https://arxiv.org/html/2411.17040v1#S5.

106. Islam, M., Akter, K., Hossain, Md. A. & Dewan, M. PD-Net: Parkinson's Disease Detection Through Fusion of Two Spectral Features Using Attention-Based Hybrid Deep Neural Network. *Information* **16**, 135 (2025).

107. Yousufi, M., Damaševičius, R. & Maskeliūnas, R. Multimodal Fusion of EEG and Audio Spectrogram for Major Depressive Disorder Recognition Using Modified DenseNet121. *Brain Sci.* **14**, 1018 (2024).

108. Early and Late Fusion for Multimodal Aggression Prediction in Dementia Patients: A Comparative Analysis. https://www.mdpi.com/2076-3417/15/11/5823.

109. Western Aphasia Battery (WAB) – Strokengine. https://strokengine.ca/en/assessments/western-aphasia-battery-wab/.

110. Wilson, S. M. *et al.* Connected speech production in three variants of primary progressive aphasia. *Brain* **133**, 2069–2088 (2010).

111. Iqbal, K. WAV - Waveform Audio File Format. https://docs.fileformat.com/audio/wav/ (2019).

112. About FFmpeg. https://www.ffmpeg.org/about.html.

113. REAPER | Audio Production Without Limits. https://www.reaper.fm/.

114. Schröter, H., Escalante-B, A. N., Rosenkranz, T. & Maier, A. DeepFilterNet2: Towards Real-Time Speech Enhancement on Embedded Devices for Full-Band Audio. Preprint at https://doi.org/10.48550/arXiv.2205.05474 (2022).

115. Défossez, A., Usunier, N., Bottou, L. & Bach, F. Demucs: Deep Extractor for Music Sources with extra unlabeled data remixed. Preprint at https://doi.org/10.48550/arXiv.1909.01174 (2019).

116. Learn from Real: Reality Defender's Submission to ASVspoof5 Challenge. https://arxiv.org/html/2410.07379v1.

117. Wuttke, F., Lyu, H., Sattari, A. S. & Rizvi, Z. H. Wave based damage detection in solid structures using spatially asymmetric encoder–decoder network. *Sci. Rep.* **11**, 20968 (2021).

118. Hansen, J. H. L., Stauffer, A. & Xia, W. Nonlinear waveform distortion: Assessment and detection of clipping on speech data and systems. *Speech Commun.* **134**, 20–31 (2021).

119. Roberts, L. Understanding the Mel Spectrogram. *Analytics Vidhya* https://medium.com/analytics-vidhya/understanding-the-mel-spectrogram-fca2afa2ce53 (2024).

120. analisiT3.pdf.

121. Mel-frequency Cepstral Coefficients (MFCC) for Speech Recognition. *GeeksforGeeks* https://www.geeksforgeeks.org/nlp/mel-frequency-cepstral-coefficients-mfcc-for-speech-recognition/ (17:11:31+00:00).

122. Kumar, K., Kim, C. & Stern, R. *Delta-Spectral Cepstral Coefficients for Robust Speech Recognition*. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings* 4787 (2011). doi:10.1109/ICASSP.2011.5947425.

123. jonatasgrosman/wav2vec2-large-xlsr-53-italian · Hugging Face. https://huggingface.co/jonatasgrosman/wav2vec2-large-xlsr-53-italian (2024).

124. Park, D. S. *et al.* SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition. in *Interspeech 2019* 2613–2617 (ISCA, 2019). doi:10.21437/Interspeech.2019-2680.

125. Sareen, V. & K.r., S. Speech Emotion Recognition using Mel Spectrogram and Convolutional Neural Networks (CNN). *Procedia Comput. Sci.* **258**, 3693–3702 (2025).

126. ElasticTransform — Torchvision main documentation. https://docs.pytorch.org/vision/main/generated/torchvision.transforms.ElasticTransform.html.

127. RandomRotation — Torchvision main documentation. https://docs.pytorch.org/vision/main/generated/torchvision.transforms.RandomRotation.html.

128. Selvaraju, R. R. *et al.* Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization. *Int. J. Comput. Vis.* **128**, 336–359 (2020).

129. Erdem (burnpiro), K. XAI Methods — Integrated Gradients. *Medium* https://medium.com/@kemalpiro/xai-methods-integrated-gradients-6ee1fe4120d8 (2022).
130. Subramanian, V., Arya, R. & Sahai, A. Generalization for multiclass classification with overparameterized linear models. Preprint at https://doi.org/10.48550/arXiv.2206.01399 (2022).
131. Chen, W. *et al.* Classifying Heart-Sound Signals Based on CNN Trained on MelSpectrum and Log-MelSpectrum Features. *Bioengineering* **10**, 645 (2023).
132. Bai, S., Kolter, J. Z. & Koltun, V. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. Preprint at https://doi.org/10.48550/arXiv.1803.01271 (2018).
133. Oyedare, T., Shah, V. K., Jakubisin, D. J. & Reed, J. H. Keep It Simple: CNN Model Complexity Studies for Interference Classification Tasks. in *IEEE INFOCOM 2023 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)* 1–6 (2023). doi:10.1109/INFOCOMWKSHPS57453.2023.10226045.
134. Abdullah, H., Ali, N. & Abdullah, N. Evaluating the Performance and Behavior of CNN, LSTM, and GRU for Classification and Prediction Tasks. *Iraqi J. Sci.* 1741–1751 (2024) doi:10.24996/ijs.2024.65.3.43.