Politecnico di Torino

Corso di Laurea Magistrale in Ingegneria Civile

Tesi di Laurea Magistrale

Emissione Acustica e Machine Learning per l'Identificazione del Danno e il Monitoraggio Strutturale

Relatori :
Prof. G. C. Marano
Prof. A. Manuello Bertetto

Candidata:
Emanuela D'Alto

Matricola:
S276543

Correlatori:
Ing. Marco Martino Rosso
Ing. Jonathan Melchiorre

Luglio 2022
A.A. 2021/2022

A chi ha creduto in me,
anche quando non l'ho fatto io

Sommario

Nel campo del monitoraggio strutturale, la tecnica delle emissioni acustiche (AE) risulta essere uno dei metodi più ampiamente utilizzati. Si tratta di una tecnica di prova non distruttiva (NDT) che impiega le onde elastiche transitorie generate dall'insorgenza e dall'evoluzione di cricche per monitorare la struttura in tempo reale. Tra le sue principali caratteristiche troviamo la capacità di identificare e localizzare il danno, attraverso la determinazione affidabile ed esatta del tempo di inizio, e la capacità di identificare la modalità di fessurazione a partire da parametri registrati. L'obiettivo del presente lavoro di Tesi è rilevare in maniera automatica il tempo di insorgenza (Onset time) dei segnali di emissione acustica con l'ausilio di algoritmi di intelligenza artificiale, più in particolare di reti neurali artificiali (ANN). Ispirate al cervello umano, quest'ultime rappresentano l'elemento centrale degli algoritmi di deep learning. Per il rilevamento automatico dell'Onset Time sono stati utilizzati due differenti approcci. Nel primo caso si è utilizzata una particolare rete neurale convoluzionale (Faster R-CNN) pre-addestrata su un dataset di grandi dimensioni. Questo ha permesso di sfruttare la tecnica del transfer learning, una valida alternativa al training da zero che permette di velocizzare i tempi di addestramento, diminuire la potenza di elaborazione richiesta e incrementare l'accuratezza della rete a fronte di un set di dati di allenamento relativamente contenuto. Per il secondo approccio si è utilizzata una rete neurale convoluzionale ricorrente (CRNN) inserita nel contesto della sound event detection (SED), classe di metodi il cui fine è quello di riconoscere in quali istanze temporali diversi suoni sono attivi all'interno di un segnale audio. Dato l'evidente parallelismo tra i segnali AE e quelli sismici, l'addestramento delle due reti è avvenuto mediante l'utilizzo di quest'ultimi. Il loro reperimento è stato reso possibile grazie a ITACA (ITalian ACcelerometric Archive), l'archivio italiano delle forme d'onda accelerometriche che contiene più di 50,000 forme d'onda. Entrambe le reti hanno permesso il raggiungimento dell'obiettivo prefissato. Tuttavia, la seconda risulta essere più vantaggiosa: essa, infatti, trattando l'individuazione del tempo di insorgenza come problema di classificazione, è in grado anche di identificare la modalità di fessurazione.

Abstract

In the field of structural monitoring, the acoustic emission technique (AE) is one of the most widely used methods. AE is a non-destructive testing (NDT) technique which investigates acoustic ultrasonic waves generated by a rapid release of energy within a material due to cracks and micro-cracks opening, analysed for structural health monitoring (SHM) purposes. Two of the most essential features of the AE technique are the ability to detect and localize the damage/crack, through a reliable and accurate determination of the onset time, and the ability to identify the cracking mode from recorded parameters. The aim of this thesis is automatically detecting the Onset Time of AE signals with the help of artificial intelligence algorithms, more specifically artificial neural networks (ANN). An ANN is a machine learning model inspired by the networks of biological neurons in human brains. Two different approaches have been used for automatic onset time detection. In the first case, using a convolutional neural network (Faster R-CNN) pretrained on a very large dataset, it was possible to use a technique called transfer learning. The main benefits of this technique, which represents a great alternative to training ANNs from scratch, include: speed up training considerably, saving of resources, improving the efficiency and removing the need for a large set of labelled training data. For the second approach a convolutional recurrent neural network (CRNN), coming from the context of the sound event detection, is used. SED is the task of recognizing the sound events and their respective temporal starting and ending time in a recording. Because of the obvious parallelism between AE signals and seismic signals, the training of the two networks has been carried out with the latter. The seismic signals dataset has been collected thanks to ITACA, Italian ACcelerometric Archive that contains more than 50,000 waveforms. The goal has been reached with both networks. However, the second is more advantageous than the first: SED treats the onset time detection as a problem of classification, so it is also able to identify the cracking mode.

Indice

Elenco delle figure

Elenco delle tabelle

Introduzione

Molte strutture in servizio soffrono di danni cumulativi causati da sovraccarichi e fessure dovute alla fatica. Al fine di valutare la durata delle strutture esistenti e ridurre i costi di manutenzione, è evidente la necessità di un sistema di monitoraggio affidabile e rigoroso per le strutture ingegneristiche [12]. Per stimare la sicurezza e le prestazioni dello stato attuale delle strutture in calcestruzzo vengono spesso applicate tecniche di valutazione non distruttiva (NDT), che non alterano il materiale e non richiedono la distruzione o l'asportazione di campioni dalla struttura in esame. Tra di esse emerge la tecnica dell'emissione acustica (AE), la quale impiega le onde elastiche transitorie generate dall'insorgenza e dall'evoluzione di cricche per monitorare la struttura in tempo reale. Il sistema di acquisizione delle emissioni acustiche rileva passivamente queste onde e le converte in segnali elettrici utilizzando trasduttori piezoelettrici montati sulla superficie. Uno dei vantaggi nell'uso della tecnica AE rispetto ad altre tecniche non distruttive è che essa è in grado di localizzare il danno e l'intera struttura può essere testata senza interromperne le prestazioni. Per la localizzazione del danno attraverso la tecnica AE risulta fondamentale la determinazione affidabile ed esatta del tempo di inizio dei segnali. Data l'enorme quantità di dati acquisiti durante un test, il rilevamento automatico dell'insorgenza diventa uno stato altamente preferibile. Pertanto, l'obiettivo del presente lavoro di Tesi è rilevare in maniera automatica il tempo di insorgenza (*Onset time*) dei segnali di emissione acustica con l'ausilio di algoritmi di intelligenza artificiale (IA). Sempre più utilizzata in tutti i domini ingegneristici, l'intelligenza artificiale ha alimentato molte visioni e speranze. Nel campo dell'ingegneria civile, l'IA ha preso di mira quei problemi che risultano influenzati da incertezze, parte inevitabile nell'ingegneria strutturale.

Nella prima parte dell'elaborato viene fatta un'introduzione all'Intelligenza Artificiale ponendo particolare attenzione al Machine Learning e al Deep Learning, tecniche che stanno emergendo sempre più come nuova classe di metodi intelligenti da utilizzare nell'ingegneria strutturale. Il secondo capitolo è dedicato alla tecnica dell'emissione acustica: partendo da un breve accenno alla *Meccanica della Frattura*, si arriva ad analizzare le principali caratteristiche di tale tecnica, ovvero la capacità di localizzare il danno e la capacità di identificare la modalità di fessurazione a partire da parametri registrati. Il terzo capitolo consiste in un approfondimento delle reti neurali artificiali

(ANN): ispirate al cervello umano, esse rappresentano l'elemento centrale degli algoritmi di deep learning. Dopo un'attenta analisi su come le ANN vengono addestrate, nella parte finale viene fatto un focus sulle reti neurali convoluzionali, oggetto dell'analisi. Gli ultimi due capitoli sono dedicati alle due metodologie utilizzate per raggiungere lo scopo sopra indicato, ovvero la Faster R-CNN e una rete neurale convoluzionale ricorrente (CRNN) inserita nell'ambito della sound event detection (SED). La struttura dei due capitoli prevede:

- un'iniziale introduzione sulla tecnica di riferimento: il transfer learning per la Faster R-CNN e la sound event detection per la CRNN;
- l'analisi della tipologia di rete e del modello utilizzati;
- l'applicazione della rete a un dataset contenente segnali sismici, dato l'evidente parallelismo tra quest'ultimi e i segnali AE. Questa scelta è scaturita dal fatto che il processo di addestramento di reti neurali artificali richiede una grande quantità di dati. La composizione di un dataset consistente è stata resa possibile grazie a ITACA, l'archivio italiano delle forme d'onda accelerometriche, il quale contiene più di 50.000 forme d'onda;

Capitolo 1

L'Intelligenza Artificiale

1.1 Definizione e cenni storici

Sebbene il termine Intelligenza Artificiale ci faccia pensare a tecnologie all'avanguardia, a robot in grado di comprendere e decidere le azioni da compiere e ad un mondo futuristico in cui macchine e uomini convivono, la realtà è che essa è già ampiamente presente nella nostra vita quotidiana e si tratta di utilizzi meno invasivi di quello che si pensa: in particolare in applicazioni comuni come gli assistenti vocali, le auto a guida autonoma, il riconoscimento facciale, i chatbot, etc..

Con il termine Intelligenza Artificiale (AI, Artificial intelligence) ci si riferisce a un ramo dell'informatica che permette la programmazione e progettazione di sistemi hardware e sistemi di programmi software atti a fornire all'elaboratore elettronico prestazioni che vengono considerate tipicamente umane [27].
Ripercorrendo un po' la storia, a segnare l'inizio dell'interesse verso questo argomento fu sicuramente la nascita dei primi calcolatori.
Le basi del concetto di Intelligenza Artificiale nascono nel 1936, anno in cui il padre dell'informatica moderna, Alan Turing, formulò l'ipotesi di una macchina in grado di svolgere qualsiasi tipo di calcolo. Tuttavia, il contributo più importante che l'ingegnere inglese diede al campo dell'AI fu lo scritto "*Computing machinery and intelligence*" pubblicato nel 1950 [13]. Di seguito si riporta l'incipit dell'articolo il quale contiene una domanda cruciale: '*I propose to consider the question, 'Can machines think?'*. Dunque, la domanda che propose Turing fu: '*Possono le macchine pensare?*'. Dare una risposta a questa domanda estremamente attuale risultava alquanto complesso. Ciò che fece Turing fu aggirare il problema proponendo un gioco chiamato 'The imitation game' (o Test di Turing). Il gioco consisteva nel valutare la capacità della macchina di avere un comportamento intelligente, inteso come "umano". Il test prevedeva di porre un giudice di fronte ad un terminale, tramite cui comunicare con due entità: un uomo e un computer. Se il giudice non riusciva a distinguere tra uomo e macchina, allora il computer aveva passato il test, e poteva essere definito "intelligente".

Grazie al lavoro di Turing, il tema dell'Intelligenza Artificiale ricevette una forte attenzione da parte della comunità scientifica, tant'è che oggi viene considerato uno dei padri fondatori di questa disciplina. Non fu però Turing a coniare il termine Intelligenza Artificiale. Infatti, la terminologia Intelligenza Artificiale fu utilizzata per la prima volta nel 1956 durante un seminario tenutosi presso il Dartmouth College di Hanover nel New Hampshire. Ad esso presero parte alcune delle figure di spicco del nascente campo della computazione dedicata allo sviluppo di sistemi intelligenti quali l'informatico statunitense John McCarthy, sviluppatore del Lisp, ossia il primo linguaggio di programmazione che per oltre trent'anni fu alla base dei software di Intelligenza Artificiale.
Gli anni successivi alla nascita dell'AI furono anni di grande fermento intellettuale e sperimentale e le aspettative sulle sue applicazioni iniziarono a crescere: nacquero così programmi in grado di dimostrare teoremi sempre più complessi. L'eccitazione iniziò a scemare nel decennio 1970-1980: le aspettative e le speranze, che erano state millantate dagli scienziati, si rivelarono troppo ambiziose. Poiché i macchinari dell'epoca non disponevano di una capacità computazionale adeguata e la scoperta che la capacità di ragionamento delle macchine si basava solo su una mera manipolazione sintattica priva di una conoscenza semantica portò alla frammentazione dell'Intelligenza Artificiale in distinte aree basate su teorie diverse (Intelligenza Artificiale Forte e Debole, analizzate nel seguito) e ad un brusco rallentamento della ricerca in questo settore.
Un nuovo impulso alla ricerca venne dal campo biologico: a partire dalla realizzazione di DENDRAL, programma in grado di ricostruire una molecola semplice a partire dalle informazioni ottenute dallo spettrometro di massa, sino ad arrivare a un algoritmo che permetteva l'apprendimento per reti neurali, già ideato alla fine degli anni Sessanta ma che non aveva trovato la massima applicazione a causa delle carenze dovute ai sistemi di apprendimento dei primi programmi di Intelligenza Artificiale.
Negli ultimi venti anni il progresso è avanzato a ritmi spaventosi, consentendo di raggiungere traguardi impensati. I motivi che hanno permesso uno sviluppo così impressionante sono principalmente due: l'aumentata potenza e capacità di calcolo degli elaboratori, che hanno reso possibili algoritmi sempre più complessi e sofisticati, e l'enorme disponibilità di dati personali e non resi fruibili dalla popolazione tramite l'utilizzo della tecnologia. Al giorno d'oggi i sistemi intelligenti sono presenti in ogni campo e l'Intelligenza Artificiale rappresenta uno dei principali ambiti di interesse della comunità scientifica informatica, con temi di ricerca come il Machine Learning, l'elaborazione del linguaggio naturale e la robotica.

1.2 Intelligenza Artificiale forte e debole

Alla base del concetto di AI vi è il desiderio dell'uomo di creare una connessione indissolubile tra automazione e ragionamento.
Se in una prima fase l'AI era associata puramente alla realizzazione di macchine e programmi in grado di fornire soluzioni di teoremi matematici più o meno complessi, in una seconda fase ci si scontrò con la volontà di ricercare soluzioni a problematiche più vicine alla realtà dell'uomo, come problemi le cui soluzioni potevano variare a seconda dell'evoluzione dei parametri in corso d'opera. L'obiettivo era, dunque, quello di cercare di riprodurre software e macchine che potessero ragionare e prendere delle decisioni in base all'analisi di differenti possibilità. Le difficoltà riscontrate nel raggiungimento di tale obiettivo diedero vita a due teorie fondamentali: Intelligenza Artificiale Forte e Debole.
Alla base dell'AI Forte vi è l'idea che un computer opportunamente programmato non sia solo una simulazione della mente, ma che possa essere una mente esso stesso, con una capacità cognitiva non distinguibile da quella umana. In altre parole, non si guarda più al computer come un mero strumento, ma come una macchina in grado di sviluppare una coscienza di sé.
I sostenitori di questa branca ritengono che sia possibile creare una macchina artificiale uguale, se non addirittura superiore, alla mente umana.
Si collocano in questo ambito i "sistemi esperti", cioè una serie di programmi che vogliono riprodurre le prestazioni e le conoscenze delle persone esperte in un determinato campo. Il sistema esperto opera in tre step distinti. Il primo consiste in regole e procedure di cui il sistema ha bisogno nel corso del suo operato. Il secondo, nonché il cuore di questi sistemi, è il motore inferenziale ossia un algoritmo che, come la mente umana, da una proposizione assunta come vera passa a una seconda proposizione. Infine, il terzo si esplica nell'interfaccia utente ove si profila l'interazione tra la macchina e l'essere umano.
Fu il filosofo statunitense John Searle ad introdurre la differenziazione tra AI forte e debole nell'articolo '*Minds, Brains and Programs*' del 1980. [14]
Searle sosteneva che non fosse possibile assimilare la mente dell'uomo al computer: ciò che un sistema informatico fa è riprodurre artificialmente gli atti umani, ma non possiamo parlare di atti propriamente umani di comprensione poiché, in quest'ultimo caso, entrano in gioco coscienza e intenzionalità che sono "fenomeni" irriducibili. Vale a dire, dato che la mente possiede intenzionalità, e il computer no, il computer non può avere una mente.
Per sostenere la sua tesi, Searle ideò un esperimento mentale noto come 'Esperimento della Stanza Cinese', di seguito riportato [15]:
≪*Si immagini di chiudere in una stanza una persona che non conosce una parola di cinese. La persona ha a disposizione due gruppi di fogli: sui fogli del primo si trova una serie di caratteri cinesi, sugli altri fogli ci sono delle istruzioni su come utilizzare*

i caratteri stessi. Il compito assegnato alla persona in questione è di produrre degli insiemi di caratteri cinesi (risposte), seguendo unicamente le istruzioni ricevute, ogni volta che riceve dall'esterno degli insiemi di caratteri cinesi (domande). Il punto fondamentale dell'esperimento è che a un cinese che ponga le domande e legga le risposte ricevute, la persona chiusa nella stanza appare come se fosse in grado di comprendere il cinese, mentre, in realtà, si limita a manipolare simboli senza significato sulla base di istruzioni.≫

In conclusione, con questo esperimento Searle vuole dimostrare come non sia possibile in nessun modo che una macchina pensi, in quanto vorrebbe dire che creare un sistema di caratteri casuali identici al cervello, cosa che risulta irrealizzabile. In sostanza, il computer per elaborare l'informazione non ha bisogno di comprendere il linguaggio o altri codici simili.

L'Intelligenza Artificiale debole, invece, si fonda sull'idea che il computer agisce e pensa come se avesse un cervello, ma non è intelligente, simula solo di esserlo. L'obiettivo è, dunque, quello di realizzare macchine in grado di svolgere una o più funzioni umane complesse senza avere coscienza delle attività svolte. In altre parole, a differenza dell'AI forte, non vi è la pretesa né di eguagliare né di superare il comportamento umano, ma solo di simularlo.

In questo caso la presenza dell'uomo è imprescindibile: sebbene la macchina svolga egregiamente il suo compito, ovvero realizzare un'intelligenza "simulata", non è in grado di pensare in maniera autonoma. Il modus operandi dell'AI debole nel fornire una risposta ad un problema è il seguente: raccogliere esperienze indagando su casi simili, confrontarle, elaborare una serie di soluzioni e poi scegliere quella più razionale e congrua, coerentemente al comportamento umano.

In conclusione, l'AI debole non ha la pretesa di comprendere totalmente i processi cognitivi umani né di definire i processi che crea come processi mentali, ma si occupa sostanzialmente di problem solving, ovvero dare risposte a problemi sulla base di regole conosciute.

1.3 Machine Learning

Con il termine Machine Learning (in italiano, apprendimento automatico) ci si riferisce a un sottoinsieme dell'Intelligenza Artificiale che si basa su sistemi in grado di apprendere dall'esperienza, svolgendo ragionamenti induttivi. La teoria di base è che i computer possono imparare ad eseguire compiti specifici senza essere programmati per farlo, utilizzando algoritmi che imparano dai dati in modo iterativo.
I termini Machine learning e AI vengono spesso utilizzati in modo interscambiabile, ma non hanno lo stesso significato: l'apprendimento automatico può considerarsi una strada per l'applicazione dell'Intelligenza Artificiale, più ampio campo di ricerca che studia lo sviluppo di sistemi Hardware e Software dotati di capacità tipiche dell'essere umano.
La definizione più citata di Machine Learning (ML) appartiene all'americano Tom Mitchell, il quale nel suo libro "Machine Learning" [16] pubblicato nel 1997 scrive:
"*Si dice che un programma apprende dall'esperienza E con riferimento a alcune classi di compiti T e con misurazione della performance P, se le sue performance nel compito T, come misurato da P, migliorano con l'esperienza E.*"
Il termine ML, in realtà, fu coniato ben prima, cioè nel 1959, dallo scienziato americano Arthur Lee Samuel. Sebbene, dunque, non si tratti di una disciplina recente, questa sta avendo solo oggi la possibilità di diventare concreta grazie a due elementi principali: l'aumentata capacità di calcolo degli elaboratori e l'enorme disponibilità di dati.
Gli algoritmi sono i motori che alimentano il Machine Learning. A seconda del tipo di algoritmo utilizzato per permettere l'apprendimento alla macchina, ossia a seconda delle modalità con cui la macchina impara ed accumula dati e informazioni, si possono suddividere tre differenti sistemi di apprendimento automatico:

- Apprendimento supervisionato (Supervised Learning) In questo caso, viene fornito un set di dati già etichettato (input di cui si conoscono già gli output) che addestri il computer permettendogli di costruire un vero e proprio database di informazioni e di esperienze. Il compito dell'algoritmo è, dunque, quello di mettere in relazione questi dati definendo un modello sulla base del quale dare un risultato corretto quando la macchina si trova di fronte ad un nuovo problema;

- Apprendimento non supervisionato o senza supervisione (Unsupervised Learning) Viene utilizzato su dati privi di etichetta (per i quali non è stato definito un output specifico) o non strutturati, quindi dati non classificati in precedenza. L'obiettivo è, dunque, quello di derivare una regola per raggruppare i casi che si presentano, individuandone una qualche struttura interna. Dovrà essere la macchina stessa, quindi, a catalogare tutte le informazioni in proprio possesso, organizzarle ed imparare il loro significato, il loro utilizzo e, soprattutto, il risultato a cui esse portano. Risulta chiaro che ci si trova in presenza di una metodologia molto

più avanzata della precedente. Essa si è potuta sviluppare con maggiore efficacia grazie alla crescita dei Big Data;

- Apprendimento per rinforzo (Reinforcement learning) Spesso usato in videogiochi e navigazione, rappresenta probabilmente il sistema di apprendimento più complesso. In questo caso, l'algoritmo conosce l'obiettivo da raggiungere e definisce il modo in cui comportarsi sulla base di una situazione che cambia, imparando attraverso la prova e l'errore. L'apprendimento avanza per "premi", definiti rinforzi e presenta tre componenti principali: l'agente (chi impara o prende decisioni), l'ambiente (tutto ciò con cui l'agente interagisce) e le azioni (cosa può fare l'agente). Quindi, l'obiettivo è quello di imparare quali sono le azioni migliori da attuare per un migliore adattamento all'ambiente circostante.

Una delle principali caratteristiche del Machine Learning è la sua stretta correlazione con altri rami dell'informatica, tanto da rendere spesso difficile comprendere il reale limite tra di essi.

1.3.1 Data Mining

Il data mining è un'area che deve gran parte della sua ispirazione e delle sue tecniche all'Apprendimento Automatico. Pertanto, sono spesso usati come sinonimi, ma in realtà sono concetti molto diversi con obiettivi diversi.
Per data mining si intende l'estrazione di informazioni di varia natura (non risapute a priori) tramite estrapolazione mirata da grandi banche dati, con il fine di migliorare la conoscenza della macchina. In altre parole, questo processo trasforma una vasta raccolta di dati grezzi in informazioni utili. Si parla sempre di una forma di apprendimento, ma limitata al solo apprendimento non supervisionato.
Sebbene l'utilizzo di queste tecniche possa sembrare simile, quello che differenzia i rami legati al Data Mining e al Machine Learning è lo scopo per il quale tali sistemi sono stati realizzati. Per entrambi l'apprendimento non supervisionato risulta essere parte integrante; entrambi rientrano nel campo della scienza dei dati e aiutano a darne un senso, il che porta a risolvere problemi complessi. Tuttavia, se il Data Mining punta esclusivamente a far migliorare la macchina tramite conoscenze sempre nuove, il Machine Learning ha come scopo quello di un apprendimento sempre più profondo, che prende in considerazione non solo la possibilità di nuove conoscenze, ma anche quella di riproduzione delle conoscenze effettuate, con il fine di trovare preziosi modelli sottostanti all'interno dei dati complessi per prevedere i risultati futuri.
A volte il Machine Learning utilizza le tecniche di Data Mining per migliorare il processo di apprendimento. Tuttavia, come emerso fino ad ora, i due ambiti sono separati tra loro.

1.3.2 Big Data

Come emerso fino ad ora, il ML richiede la presenza di grandi quantità di dati necessari a realizzare un processo di apprendimento: più i modelli sono esposti ai dati, più sono in grado di adattarsi in modo autonomo.
Con il termine Big Data si indica una grande quantità di dati ed informazioni che vengono acquisite e gestite quotidianamente da società o enti. La loro elevata disponibilità, al giorno d'oggi, è resa possibile grazie al basso costo e alla miniaturizzazione dei dispositivi di acquisizione elettronici, nonchè allo sharing dei dati tramite internet. Sebbene non siano necessari per un utilizzo delle tecniche di apprendimento automatico, i Big Data possono contribuire a migliorare la precisione dei modelli. In altre parole, rappresentano un elemento fortemente abilitante per questo tipo di progettualità.
Al tempo stesso, però, è proprio la gestione dei Big Data a rimanere la parte più complessa del processo poiché più che l'entità di questi dati ciò che attira l'attenzione è il loro utilizzo, ovvero come essi possano essere analizzati. È dunque comprensibile come sia indispensabile la capacità di gestire questi dati, in quanto non avrebbero alcun valore se non fosse possibile analizzarli ed estrapolare informazioni.

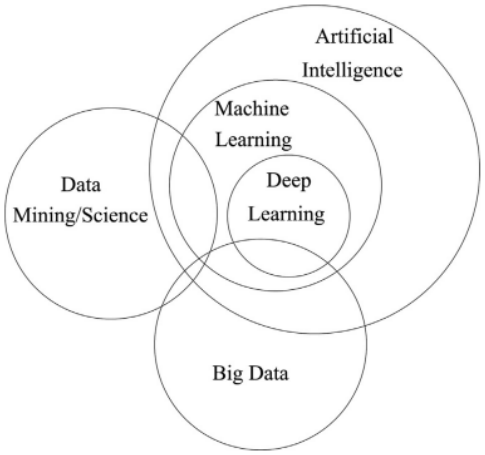

Figura 1.1. Illustrazione dell'interrelazione di diverse tecniche computazionali intelligenti [1]

1.4 Deep Learning e Reti Neurali

1.4.1 Deep Learning

Con il termine Deep Learning (in italiano, apprendimento approfondito) ci si riferisce a una sottocategoria del Machine Learning, la quale non fa altro che creare modelli di apprendimento su più livelli. Il Deep Learning costituisce una delle principali fonti di successo per l'ambito dell'Intelligenza Artificiale e sta compiendo passi da gigante, ottenendo risultati che, fino a qualche decennio fa, erano pura utopia [27]. Tale successo è dovuto alle numerose conquiste in campo informatico, relative soprattutto alla sfera dell'hardware. Come emerso fino ad ora, di fondamentale importanza risulta portare il calcolatore a fare esperienza su un quantitativo sempre maggiore di dati sensibili. Oggi, grazie all'introduzione delle GPUs, ovvero nuove unità che concorrono all'elaborazione dati, il processo di addestramento, che in passato richiedeva tempi abbastanza elevati, è diventato molto più snello. Un altro importante aiuto è derivato dalla facilità di trovare numerose collezioni di dati (dataset), fondamentali per allenare il sistema.
Per comprendere il raggio d'azione del Deep Learning, partiamo dal concetto di neurone: cuore del sistema nervoso umano, esso ne contiene più di 100.000 e sono fondamentali per ricevere e trasmettere segnali. Il neurone umano è il paradigma computazionale che nutre il Deep Learning e lo fa attraverso le famose Reti Neurali Artificiali.

1.4.2 Definizione e storia delle Reti Neurali Artificiali

In inglese definite Artificial Neural Network (ANN, o anche semplicemente Neural Network (NN)), le Reti Neurali sono modelli matematici costituiti da neuroni artificiali che, nell'intento di replicarlo artificialmente, emulano il funzionamento del cervello umano, per raggiungere prestazioni cognitive che in qualche modo gli si avvicinino.
Nel 1943 venne proposto il primo modello teorico di un rudimentale neurone artificiale dalla coppia di scienziati McCulloch e Pitts. I due descrissero un apparato in grado di ricevere n dati binari in ingresso in ognuno dei suoi elementi, a cui segue un singolo dato in uscita per ciascuno.
Nel 1958 lo psicologo Franck Rosenblatt presentò al mondo accademico il famoso Perceptron (percettrone), primo schema di Rete Neurale Artificiale. Esso è pensato per il riconoscimento e la classificazione di forme e consiste in un'entità con un ingresso, un'uscita e una regola di apprendimento basata sulla minimizzazione dell'errore. Il percettrone costituisce un progresso decisivo rispetto al modello binario di McCulloch e Pitts, in quanto i suoi pesi sinaptici (un peso indica la forza di una connessione fra due nodi) sono dinamici e quindi la macchina è in grado di apprendere. Si tratta di un modello di tipo feedforward: gli impulsi si propagano in un'unica direzione, in avanti, limitando di molto il suo campo di applicazione.
L'opera di Rosenblatt suscitò un vivo interesse e notevoli aspettative nella comunità

scientifica, fino a quando nel 1969 Marvin Minsky e Seymour A. Papert misero in discussione il suo percettrone mostrandone i limiti operativi: dimostrarono l'impossibilità di risolvere per questa via tutti quei problemi non caratterizzati da separabilità lineare delle soluzioni.
Nel mondo scientifico si diffuse la sfiducia nelle Reti Neurali, le quali tornarono alla ribalta grazie al concetto di Rete Neurale Multistrato (MLP-Multilayer Perceptron) dove ogni strato di nodi è completamente connesso con quello successivo. Al suo interno, fra i nodi di input e quello di output si trova uno strato hidden, dove avviene l'elaborazione delle informazioni provenienti dallo strato di input, che poi vengono inviate al nodo di output. Si tratta, dunque, di una rete feedforward non lineare, in cui le connessioni in ingresso e in uscita da ogni singolo nodo sono multiple. Il contesto matematico per addestrare le reti MLP fu stabilito dal matematico americano Paul Werbos nella sua tesi di dottorato del 1974.
Nel 1986 David E. Rumelhart, G. Hinton e R. J. Williams ripresero e perfezionarono il lavoro di Werbos, elaborando il celebre Error Back-Propagation (in italiano, retropropagazione dell'errore), uno dei metodi più noti ed efficaci per l'addestramento delle reti MLP. Con questo algoritmo entriamo nel presente, essendo tuttora utilizzato. L'EBP permette di perfezionare in stadi successivi l'apprendimento automatico di una Rete Neurale e lo fa nel seguente modo: l'algoritmo confronta il valore in uscita del sistema con il valore desiderato. Sulla base della differenza così calcolata (errore), l'algoritmo torna indietro nel percorso modificando i pesi sinaptici della rete neurale e facendo convergere progressivamente il set dei valori di uscita verso quelli desiderati. Più sono gli esempi (e migliori sono) che vengono "digeriti" dalla rete e propagati all'indietro e maggiore sarà la probabilità che la rete effettui le associazioni giuste.
Con i MLP e l'EBP il machine learning trova ora alcuni campi di applicazione pratica, ma soltanto negli ultimi dieci anni si è riusciti a dimostrare la loro utilità in un'ampia gamma di applicazioni.
Intanto vengono implementate anche reti neurali con architetture feedback, in cui le informazioni fra nodi viaggiano in qualunque direzione: in avanti, all'indietro e fra nodi di una stessa fila. Il campo delle applicazioni si amplia ulteriormente.

1.4.3 Funzionamento delle Reti Neurali

Per comprendere il funzionamento di una Rete Neurale Artificiale partiamo da quello di una Rete Neurale Biologica.
Ogni volta che il cervello riceve uno stimolo sensoriale un gruppo di neuroni riceve un segnale, chiamato potenziale d'azione. Se questo risulta essere abbastanza forte, viene propagato ai neuroni vicini tramite dei canali chiamati sinapsi. Questo processo si ripete a cascata (neuroni attivano altri neuroni) fino a quando il segnale si esaurisce. I neuroni che si attivano insieme si legano, formando complessi reticoli che sono proprio le reti neurali, sede di tutta la nostra conoscenza.

Parallelamente, una rete neurale artificiale è costituita da elementi di elaborazione interconnessi, chiamati nodi o neuroni. Questi ricevono in ingresso degli stimoli e li elaborano, moltiplicandoli per un opportuno valore detto peso, il quale fornisce una misura di quanto "conta" tale input nel neurone.

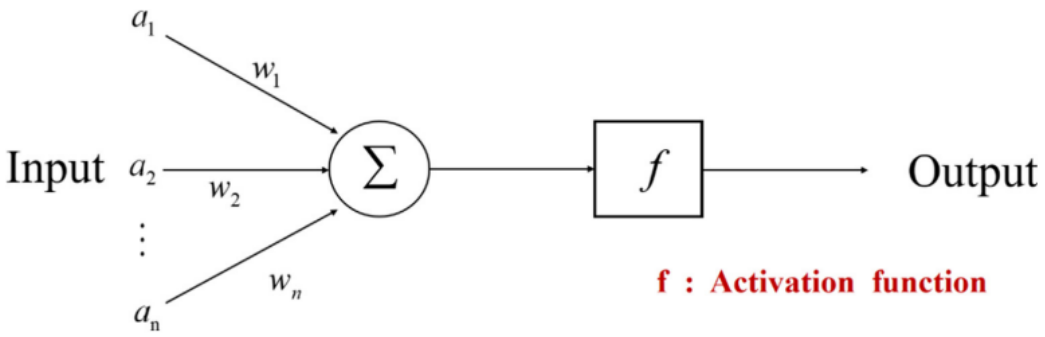

Figura 1.2. Concetto base di un singolo neurone [2]

Il risultato delle moltiplicazioni viene sommato e sottoposto ad una funzione di attivazione (Activation function). Se la somma supera una certa soglia il neurone si attiva producendo un output secondo la seguente formula:

$$Output = f(\sum_{i=1}^{n} w_i a_i) = f(w_1 a_1 + w_2 a_2 + \cdots + w_n a_n) \tag{1.1}$$

Se nei neuroni biologici il potenziale d'azione viene trasmesso una volta che la differenza di potenziale alle membrane supera una certa soglia, per i neuroni "artificiali" il comportamento della risposta è determinato dalla funzione di attivazione. Ma perché risulta necessario applicarla?
Un uso importante di questa funzione è di mantenere l'uscita limitata a un intervallo particolare; inoltre, una Rete Neurale senza funzione di attivazione equivale semplicemente a un modello di regressione, che cerca di approssimare la distribuzione dei dati con una retta. Tuttavia, lo scopo delle Reti Neurali è quello di essere in grado di approssimare qualsiasi funzione rendendo, dunque, necessario introdurre un fattore di non linearità, da qui la funzione di attivazione. Tra le più comuni troviamo:

- Funzione "soglia" (Threshold Function), il cui meccanismo risulta essere molto semplice: la funzione restituisce 1 nel caso in cui la somma pesata dei segnali in input è maggiore o uguale a zero, e 0 nei restanti casi. Tuttavia, uno dei suoi limiti risiede nel fatto che non è differenziabile nel punto in cui cambia valore. La derivata è, però, fondamentale nel deep learning, in quanto determina la direzione verso cui orientarsi per gli aggiustamenti ai valori;

- Funzione ReLU (REctifier Linear Unit), divenuta ultimamente molto utilizzata in quanto si tratta di una funzione facile da calcolare. Questa funzione restituisce

0 qualora la somma pesata dei segnali in input sia minore o uguale a zero, oppure $\sum wa$ negli altri casi. In questo caso, il codominio della funzione spazia da 0 ad infinito. La derivata è molto semplice: per tutti i valori negativi è uguale a zero, mentre per quelli positivi è uguale a 1. Nel punto angoloso nell'origine, invece, la derivata è indefinita ma viene comunque impostata a zero per convenzione;

- Funzione sigmoide, la quale ricalca la funzione soglia con la differenza che il passaggio da 0 a +1 è più graduale (con un andamento a forma di s appunto) rendendola differenziabile;

- Funzione Tangente Iperbolica, che è simile alla sigmoide dalla quale differisce per il semplice fatto che il suo codominio va da -1 a +1;

Le ultime due funzioni sono interessanti in quanto permettono al neurone di avere un output continuo.
Sino a qui, si è analizzato il funzionamento di un singolo neurone. Per formare una Rete Neurale, i singoli neuroni lavorano in parallelo e vengono collegati tra loro per mezzo di connessioni pesate, chiamate 'sinapsi'. In linea generale, le Reti Neurali si compongono di tre tipi di strati, capaci di coinvolgere migliaia di neuroni e decine di migliaia di connessioni: lo strato di ingresso (Input layer), gli strati "nascosti" (Hidden layers) e lo strato di uscita (Output layer).

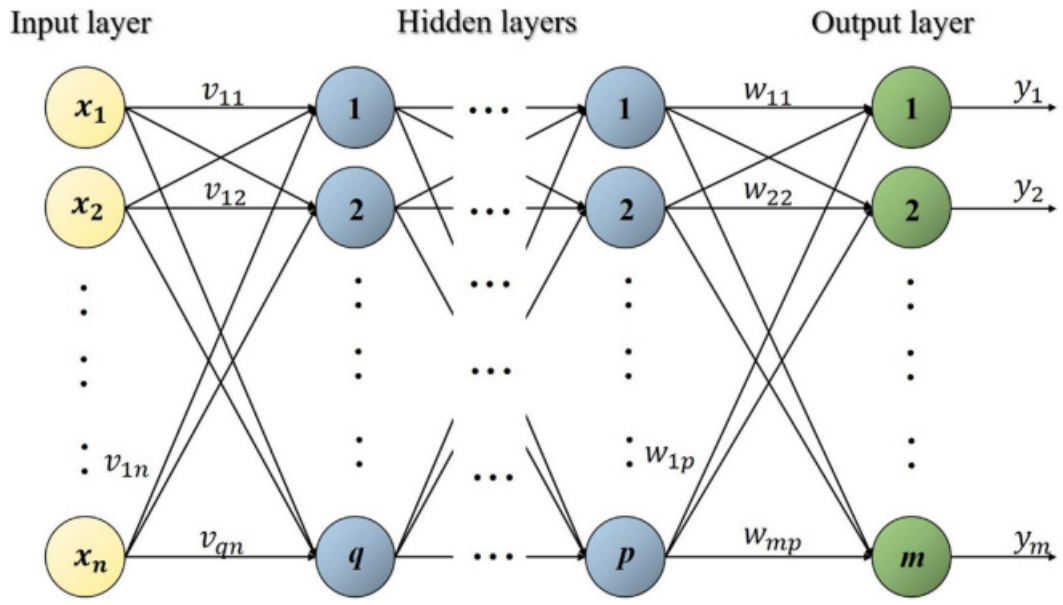

Figura 1.3. Tipica struttura di una ANN [2]

Il primo strato è quello che ha il compito di ricevere segnali e dati provenienti dall'esterno sotto forma di input, elaborarli e passare le informazioni ottenute allo strato di neuroni successivo; il secondo, che può essere formato da uno o più strati nascosti, ha

in carico il processo di elaborazione vero e proprio; infine, il terzo raccoglie i risultati dell'elaborazione dello strato H, i quali vengono adattati alle richieste del blocco successivo della rete neurale, fino ad arrivare alla presentazione della soluzione definitiva al problema sottoposto. Più complesso è il problema da risolvere, più strati sono necessari.
Esistono diverse tipologie di reti neurali, che possono essere distinte per tipo e per algoritmo di apprendimento utilizzato. Il tipo di una rete neurale indica come i neuroni dei vari strati della rete possono essere collegati tra loro. Da qui la suddivisione tra:

- Rete feed forward, modello più semplice che non prevede cicli, ovvero l'informazione passa dall'input all'output. In questo caso, ogni neurone di un livello riceve input solo dai neuroni del livello precedente e propaga gli output solamente verso i neuroni dei livelli successivi. In altre parole, in questo tipo di reti non sono possibili autocollegamenti o connessioni con i neuroni del proprio livello e, di conseguenza, l'output di qualsiasi layer non influisce su quello stesso layer. Una conseguenza immediata di ciò è rappresentata dal fatto che la rete risponde sempre nello stesso modo se sollecitata con gli stessi input. Rientrano in questa categoria le Reti Neurali Multistrato (MLP-Multilayer Perceptron), modello di rete neurale più ampiamente studiato e utilizzato;

- Rete Feedback (o ricorrente), la quale consente anche connessioni tra i neuroni dello stesso strato, introducendo così loop nella rete. Questa tipologia di rete può diventare estremamente complicata. I calcoli derivati da input precedenti vengono reimmessi nella rete, il che conferisce loro una sorta di memoria. Le reti feedback sono dinamiche; il loro "stato" cambia continuamente fino a raggiungere un punto di equilibrio, nel quale rimangono fino a quando l'ingresso non cambia. Rientra in questa categoria la Rete di Hopfield, costituita da un insieme di neuroni in cui ogni neurone è collegato a tutti gli altri e non c'è differenziazione tra neuroni di input e di output.

Un altro modo di classificare le ANN è quello basato sull'algoritmo di apprendimento: affinché una rete risulti performante è necessario "addestrarla". Nelle reti artificiali ovviamente il processo di apprendimento automatico è semplificato rispetto a quello delle reti biologiche. Non esistono analoghi dei neurotrasmettitori, ma lo schema di funzionamento è simile.
Il tema dell'apprendimento è collegato al Machine Learning, inteso come algoritmi che utilizzano metodi matematico-computazionali per apprendere informazioni dall'esperienza (quindi in modo automatico e adattivo). Anche nell'addestramento delle reti neurali, dunque, è possibile distinguere le tre categorie di algoritmi di apprendimento enunciate nel Machine Learning:

- Apprendimento Supervisionato (Supervised Learning)

- Apprendimento non supervisionato (Unsupervised Learning)

- Apprendimento per rinforzo (reinforcement learning)

Per concludere, cos'hanno dunque di così speciale le Reti Neurali?
I vantaggi risiedono nell'elevato parallelismo, grazie al quale si possono processare grandi moli di dati in tempi relativamente rapidi rispetto ai calcolatori tradizionali, in cui ciascun dato viene elaborato individualmente e in successione. Inoltre, sono poco sensibili al rumore, quindi sono in grado di operare, in molti casi, in modo corretto nonostante input imprecisi o incompleti.
D'altro canto, un grande limite risiede nell'incapacità di spiegare il comportamento delle reti neurali. Si parla, infatti, di funzionamento a black box: le reti neurali sono in grado di fornire output corretti, o sufficientemente corretti, ma non permettono di esaminare i singoli stadi di elaborazione che li determinano.

1.5 L'Intelligenza Artificiale nell'Ingegneria Strutturale

Negli ultimi anni, c'è stato un crescente interesse in tutti i domini ingegneristici per l'uso dell'Intelligenza Artificiale, la quale ha alimentato molte visioni e speranze. Tuttavia, il settore delle costruzioni è tra quelli rimasti più indietro nell'adozione (o quantomeno nella sperimentazione) di soluzioni di IA. Una lentezza ravvisata anche dalla società di consulenza *McKinsey & Company* che in un report del 2018 [17] ha spiegato:
≪Il settore ingegneristico e delle costruzioni ha raggiunto un valore annuale superiore ai 10 trilioni di dollari. Ma mentre i suoi clienti sono sempre più sofisticati, rimane profondamente lontano dai più moderni processi di digitalizzazione. Inoltre, il settore investe attualmente solo l'1% del suo giro d'affari nelle nuove tecnologie, una proporzione ben inferiore rispetto a quanto accade in altri comparti, come ad esempio il manifatturiero≫.
Nel settore dell'ingegneria civile, l'Intelligenza Artificiale ha preso di mira quella classe di problemi che sfida la soluzione tramite tecniche computazionali tradizionali, cercando di catturare l'essenza della cognizione umana al livello più alto.
Più in particolare, nel campo dell'ingegneria strutturale ci sono numerosi problemi che sono influenzati da incertezze, ad esempio quelli relativi alla progettazione, all'analisi, al monitoraggio delle condizioni, alla gestione delle costruzioni, ecc. Tali problemi richiedono calcoli matematici, fisici e meccanici per essere risolti e la loro soluzione dipende fortemente dall'esperienza dei praticanti. In altre parole, le incertezze sono una parte inevitabile dei problemi di ingegneria strutturale. Rispetto ai metodi tradizionali, l'IA offre vantaggi per affrontare problemi legati alle incertezze, diventando un aiuto efficace per risolvere problemi così complessi. Inoltre, l'uso di metodi di Intelligenza Artificiale può anche tradursi in significativi risparmi di tempo e costi, oltre ad aumentare l'efficienza computazionale in molte attività di ingegneria strutturale.
Tra le diverse tecniche di IA, Machine Learning (ML), Pattern Recognition (PR), e Deep Learning (DL) stanno emergendo sempre più come nuova classe di metodi intelligenti da utilizzare nell'ingegneria strutturale. [1]

1.5.1 Pattern recognition

Con il termine Pattern Recognition (in italiano, riconoscimento dei modelli) si indica una branchia dell'Intelligenza Artificiale che si occupa del riconoscimento dei pattern e ha come obiettivo il classificare gli oggetti in un numero di categorie o classi. Si tratta di una disciplina strettamente correlata al machine learning, al data mining e alla scoperta di conoscenza.
Un pattern è un campione di dati che veicola un'informazione utile. Nell'IA il pattern è un sinonimo di schema, modello, esempio, ed è rappresentato da un insieme di caratteristiche. Esistono diverse tipologie di pattern: pattern numerici (vettori composti da valori numerici continui o discreti rappresentanti valori, proprietà o caratteristiche misurabili), pattern categorici (proprietà e caratteristiche qualitative di un oggetto, non

mappabili in modo numerico) e pattern sequenziali (sequenze di dati a lunghezza fissa o variabile).
Come nel ML, i metodi di PR possono essere classificati in due categorie principali:

- PR supervisionato, il quale si riferisce alla condizione in cui è disponibile una serie di campioni di addestramento etichettati;
- PR senza supervisione, quando non ci sono informazioni precedenti sulle etichette della classe e i dati di allenamento non sono etichettati.

Per stabilire i confini decisionali tra le classi di pattern vengono utilizzati i concetti della teoria delle decisioni statistiche. Esistono diverse tecniche di Pattern Recognition:

- Classificazione, nella quale si suddividono i pattern in due (binary classification) o più classi (multi-class classification) raggruppando quelli che possiedono una o più proprietà in comune. Una classe è caratterizzata da una funzione in grado di definire uno spazio di appartenenza in modo chiaro e inconfutabile. Il sistema di apprendimento deve, dunque, produrre un modello in grado di assegnare ad un input una o più classi tra quelle disponibili. La classificazione utilizza tipicamente tecniche di apprendimento supervisionato;
- Regressione, concettualmente simile alla classificazione con la differenza che l'output ha un dominio continuo e non discreto. La regressione crea una funzione in grado di approssimare dei dati in output a partire dai dati in input. Anch'essa è tipicamente affrontata con l'apprendimento supervisionato;
- Clustering, tecnica per individuare dei gruppi (cluster) di pattern con caratteristiche e proprietà simili tra loro. A differenza della classificazione, in questo caso le classi non sono ancora definite perché mancano le relazioni di causalità. Tuttavia, i dati mostrano comunque delle similitudini da non trascurare. Una volta approfondita la causa della regolarità dei pattern in un cluster, si può definire la classe. La natura stessa dei problemi appartenenti a questa categoria li rende tipicamente dei task di apprendimento non supervisionato.

Durante l'ultimo decennio, c'è stato un crescente interesse per l'applicazione del riconoscimento di modelli (PR) all'ingegneria strutturale per scopi quali monitoraggio della salute strutturale (SHM), ingegneria sismica e progettazione sismica, affidabilità strutturale, identificazione strutturale e valutazione delle prestazioni.

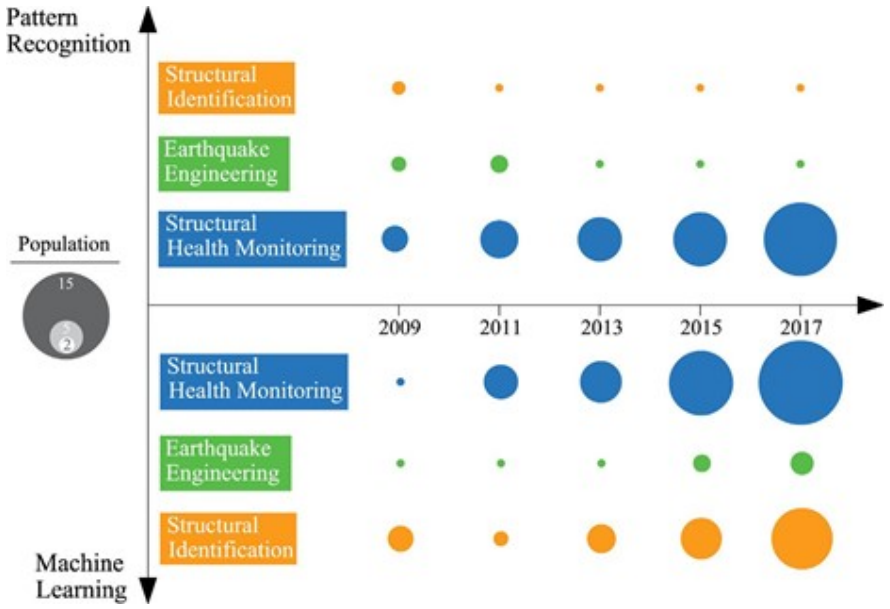

Figura 1.4. Pubblicazioni di ricerca sull'uso dell'apprendimento automatico e del riconoscimento di modelli [1]

Come emerge dal grafico contenente le pubblicazioni di ricerca in questo ambito, l'uso più comune di PR nell'ingegneria strutturale è stato per l'SHM.
Nell'ingegneria civile, con il termine "Monitoraggio della salute strutturale" (Structural Health Monitoring, SHM) si indica il processo di attuazione di una strategia di rilevamento dei danni. In termini più generali, il danno può essere definito come modifiche introdotte in un sistema che influiscono negativamente sulle sue prestazioni attuali o future. L'SHM prevede l'osservazione del sistema tramite misurazioni dalle quali vengono estratte delle caratteristiche che, in seguito ad analisi, permettono di determinare lo stato attuale di salute del sistema. In altre parole, mediante questo processo viene periodicamente controllata la capacità della struttura di espletare la propria funzione nonostante il degrado e l'invecchiamento relativi all'operatività e all'ambiente circostante.
Una procedura SHM inserita in un contesto di PR statistico viene implementata in quattro fasi [18], come riportato nel grafico sottostante:

1. Valutazione operativa, la quale inizia a stabilire i limiti su ciò che verrà monitorato e su come verrà realizzato il monitoraggio, adattando il processo alle caratteristiche che sono uniche per il sistema monitorato;

2. Acquisizione dati e pulizia, fase che comporta la selezione dei tipi di sensori da utilizzare, il modo in cui devono essere posizionati, la determinazione del numero di sensori da utilizzare e la definizione dell'hardware di acquisizione/ memorizzazione/trasmissione dei dati. In questa fase giocano un ruolo importante le considerazioni economiche e la frequenza con cui devono essere raccolti i dati. La

pulizia dei dati è il processo di scelta selettiva dei dati da accettare o rifiutare per la selezione delle caratteristiche;

3. Selezione delle caratteristiche, le quali verranno utilizzate per il rilevamento dei danni e sono, in genere, specifiche dell'applicazione. Dato che l'implementazione operativa e le tecnologie di misurazione diagnostica necessarie per eseguire il monitoraggio della salute strutturale producono in genere una grande quantità di dati, risulterà necessariamente la condensazione dei dati. In questa fase, dunque, si identificano numerose caratteristiche, le quali vengono assemblate in un vettore di "features". In generale, è desiderabile un vettore di caratteristiche a bassa dimensione;

4. Sviluppo di modelli statistici, fase che riguarda l'implementazione degli algoritmi che operano sulle caratteristiche estratte per quantificare lo stato di danneggiamento della struttura. Lo stato di danno di un sistema può essere descritto come un processo in cinque fasi: esistenza del danno, la sua posizione, la tipologia, l'entità e infine previsioni sulla vita utile. I modelli statistici sono utilizzati per rispondere in modo quantificabile a queste domande, le cui risposte nell'ordine presentato rappresentano una crescente conoscenza dello stato di danno. Infine, una parte importante del processo di sviluppo del modello statistico è il test di questi modelli su dati reali per stabilire la sensibilità delle caratteristiche selezionate al danno e per studiare la possibilità di false indicazioni di danno. Queste si dividono in due categorie: indicazione di danno falso positivo (indicazione di danno quando non è presente) e indicazioni di danno falso-negativo (nessuna indicazione di danno quando il danno è presente).

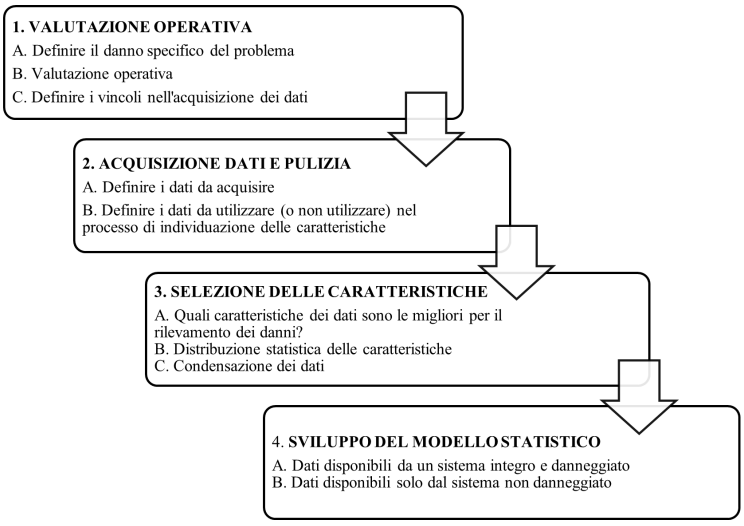

Figura 1.5. Diagramma di flusso per l'implementazione di un programma di monitoraggio della salute strutturale

1.5.2 Machine Learning

I metodi di apprendimento automatico (ML) sono stati sempre più adottati nell'ultimo decennio per modellare problemi del mondo reale riguardanti l'ingegneria strutturale. Ciò è dovuto alla loro enorme capacità di catturare relazioni tra dati di input e output che sono non lineari o complicati da formulare matematicamente. In generale, i metodi ML sono stati utilizzati per l'identificazione di danni (SHM), l'ottimizzazione, la valutazione delle prestazioni, l'affidabilità strutturale e l'identificazione dei parametri strutturali. Tra questi, l'SHM e la modellazione delle proprietà del calcestruzzo sono gli usi che hanno ottenuto maggiore attenzione nell'ultimo decennio. Come discusso in precedenza, l'SHM ha come obiettivo quello di determinare se una struttura sta funzionando come previsto o se c'è qualche anomalia nel suo comportamento rispetto alla condizione normale, permettendo, eventualmente, di rilevare l'esistenza, la posizione e la gravità del danno.
La presenza del danno può essere identificata attraverso due approcci:

- Model driven approach (approccio basato sul modello), il quale utilizza un modello numerico della struttura, ad esempio basato sul metodo degli elementi finiti (FE), che mette in correlazione le incongruenze tra i dati misurati e quelli generati dal modello per il rilevamento dei danni. Tuttavia, questo metodo presenta diverse carenze. Innanzitutto, l'approccio è inefficiente dal punto di vista computazionale perché richiede un'analisi iterativa di un modello di simulazione al computer. In secondo luogo, i risultati ottenuti dalla simulazione potrebbero non essere sufficientemente accurati per una valutazione precisa della struttura;

- Data driven approach (approccio basato sui dati), in cui il modello viene creato attraverso l'apprendimento acquisito dai dati misurati. Il danno può quindi essere rilevato effettuando un confronto tra i dati misurati e un modello. Infatti, un modello basato sui dati utilizza le informazioni dai dati dei sensori raccolti in precedenza (ad es. dati di addestramento). Tuttavia, gli approcci basati sui dati sono utili se esistono grandi volumi di dati e le caratteristiche fisiche della struttura sono sconosciute o complicate da modellare. Un approccio basato sui dati adotta comunemente tecniche di riconoscimento di modelli (PR) e apprendimento automatico (ML).

Oggi, per raggiungere i massimi livelli di sicurezza, i sistemi di monitoraggio sono chiamati a evolversi. La parola d'ordine è diventata "controllo attivo": alla tradizionale manutenzione correttiva deve preferirsi la più affidabile manutenzione predittiva. Tutto ciò è possibile grazie all'ampia disponibilità di tecnologie per la raccolta, l'archiviazione e l'analisi di dati, che sfruttano i progressi del machine learning. L'implementazione di sistemi di ML consente di automatizzare i processi di Structural Health Monitoring (SHM) con un duplice vantaggio:

- ridurre la dipendenza dal fattore umano, svincolandosi dalla soggettività intrinseca delle tradizionali procedure di ispezione e analisi difettologiche;
- gestire una base dei dati teoricamente infinita migliorando progressivamente, tramite un allenamento continuo, le proprie capacità di damage detection e predittività.

Infine, il potenziale degli algoritmi ML è stato sfruttato per modellare le proprietà meccaniche del calcestruzzo, quali resistenza alla compressione, resistenza alla trazione, resistenza al taglio e modulo elastico. Dato che le proprietà meccaniche del calcestruzzo hanno forti relazioni non lineari tra i costituenti e le caratteristiche del materiale su macroscala, lo sviluppo di modelli affidabili è interessante per esplorarle in modo da ottimizzare tempi e costi associati alle prove sui materiali.

In conclusione, si è sottolineato come PR e ML sono principalmente utilizzati dalla comunità di ingegneria strutturale per interpretare i dati dei sensori nel monitoraggio della salute strutturale. Inoltre, la precedente trattazione ha rivelato che le tecniche algoritmiche ML e PR hanno la capacità di apprendere complicate interrelazioni tra i parametri contribuenti, e quindi consentono di risolvere una varietà di problemi che sono difficili, o non possibili, da risolvere con i metodi tradizionali. Ad esempio, il rumore di misurazione, gli errori di modellazione, gli effetti ambientali, ecc. sono fattori inevitabili che potrebbero influenzare significativamente la disponibilità dei dati. È quindi essenziale utilizzare metodi di intelligenza artificiale in grado di interpretare efficacemente dati incompleti e rumorosi e valutarne le prestazioni sotto queste influenze.

Capitolo 2

L'emissione acustica

Cricche e difetti di varia natura possono influenzare in modo devastante le prestazioni di componenti e strutture a tal punto che la loro individuazione è parte essenziale del controllo di qualità in tutti i campi dell'ingegneria.
Con Emissione acustica (Acoustic Emission, AE) si indica una tecnica utilizzata per il monitoraggio strutturale che rientra nelle prove di tipo non distruttivo (NDT), ossia quelle prove che non alterano il materiale e non richiedono la distruzione o l'asportazione di campioni dalla struttura in esame. Si tratta, inoltre, di una tecnica di monitoraggio passivo, ovvero non è necessario fornire direttamente alla struttura monitorata energia dall'esterno, ma viene utilizzata l'energia dalla stessa fonte di danno. Per approfondire meglio quest'ultimo concetto, di seguito viene fatto un breve accenno alla *Meccanica della Frattura*. [3]

2.1 Introduzione alla Meccanica della Frattura

I materiali strutturali vengono tradizionalmente catalogati, in base alle caratteristiche della curva tensione-deformazione $\sigma - \varepsilon$, in due distinte categorie:

- materiali duttili, i quali mostrano ampi tratti non lineari nel diagramma $\sigma - \varepsilon$ prima di pervenire alla rottura e presentano comportamenti simili a trazione e compressione;

- materiali fragili, i quali si rompono in modo improvviso quando la risposta è ancora sostanzialmente elastica e lineare e presentano un rapporto tra resistenza a trazione e resistenza a compressione di molto inferiore rispetto all'unità (in alcuni casi $10^{-1} \div 10^{-2}$).

La distinzione tra materiali duttili e fragili non è sempre marcata in modo definito, in quanto la duttilità del materiale dipende dalla temperatura ambientale e dalla dimensione dell'elemento strutturale. Di conseguenza, la duttilità cessa di essere una proprietà del materiale e diventa una proprietà dell'intera struttura.

Una prova di trazione su di un campione di calcestruzzo condotta a carico controllato mostra una risposta approssimativamente elastica-lineare e poi, all'improvviso, una immediata caduta del carico stesso, che corrisponde alla repentina formazione di una fessura. Tuttavia, poiché le attuali tecniche consentono di pilotare la deformazione, si evidenzia la curva di risposta post-rottura del materiale cementizio.

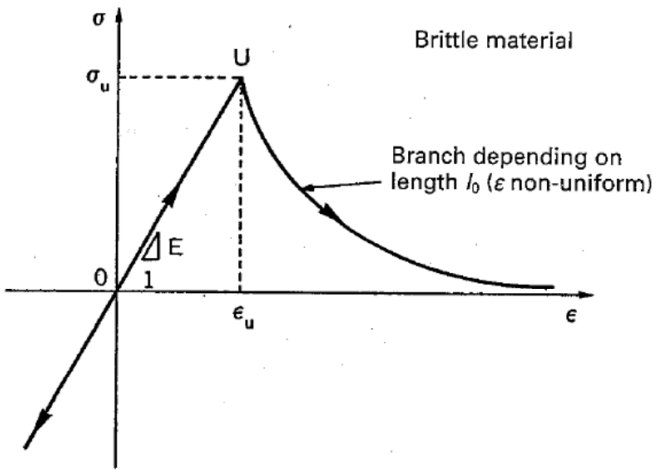

Figura 2.1. Diagramma $\sigma - \varepsilon$ con incrudimento negativo [3]

Ci si è resi così conto dell'esistenza di un esteso ramo di incrudimento negativo (in inglese, softening) e della possibilità di dissipare, da parte del materiale, una notevole quantità di energia per unità di volume. Tale energia è rappresentata dall'area sottesa dalla curva $\sigma(\varepsilon)$. Di recente si è dimostrato che l'energia in realtà non è dissipata per unità di volume, bensì è dissipata su una banda localizzata, la quale diventa in seguito una fessura. Se si riportano le curve di risposta sul piano F-Δl si ottiene il seguente grafico:

Da esso si vede che all'aumentare della lunghezza l_0 del provino, si ottengono tratti

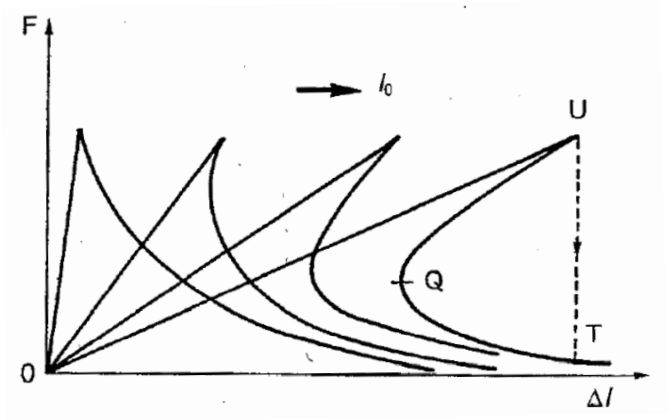

Figura 2.2. Fenomeno dello "snap-back", diagramma forza-allungamento [3]

elastici a rigidezza calante e tratti "softening" a pendenza negativa crescente e, oltre un cero limite, a pendenza positiva.
Il softening a pendenza positiva rappresenta un fenomeno inquadrabile nell'ambito della *Teoria delle Catastrofi*. Se, infatti, il processo di carico è pilotato dalla dilatazione convenzionale ε, o dall'allungamento Δl, una volta raggiunto il punto U, si ha una caduta verticale del carico sino ad incontrare il tratto "softening" inferiore. Il tratto UQT viene in questo modo ignorato e diventa virtuale. Per rilevarlo sperimentalmente, è necessario pilotare il processo di carico mediante l'apertura w della fessura. L'instabilità sopra descritta è detta snap-back. Tutti i materiali relativamente fragili presentano una brusca caduta del carico quando il comportamento globale del campione è ancora elastico lineare.
Dunque, quando nel grafico carico-spostamento si verificano gli snap-back, l'energia totale rilasciata è data dalla somma di due contributi:

$$Energia\,rilasciata(R) = Energia\,assorbita(D) + Energia\,emessa(E) \tag{2.1}$$

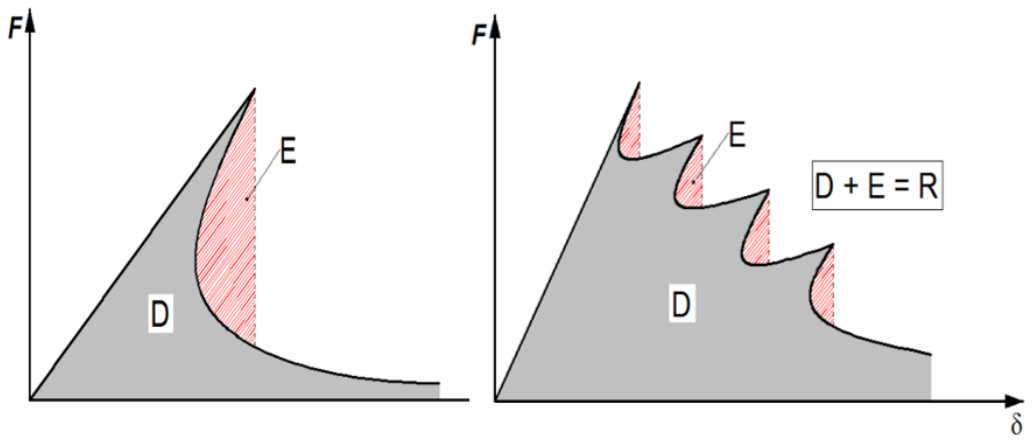

Figura 2.3. Grafico carico-spostamento con uno o più snap-back [3]

Mentre il primo contributo rappresenta l'energia che viene assorbita dalla frattura per poter avanzare, il secondo rappresenta un'energia che può essere emessa sotto forma di energia cinetica oppure acustica. Quest'ultima ha permesso lo sviluppo della tecnica di monitoraggio delle emissioni acustiche.

2.2 Tecnica e sistemi di acquisizione dati

Il monitoraggio di una struttura mediante la tecnica delle Emissioni Acustiche permette di rilevare l'insorgenza e l'evoluzione di cricche, le quali possono essere indotte da forze esterne (carico meccanico) o condizioni (invecchiamento, temperature).
Ciò è reso possibile dal fatto che la fessurazione è accompagnata dall'emissione di onde elastiche transitorie generate a causa del rapido rilascio di energia da una sorgente localizzata all'interno del materiale. Il sistema di acquisizione delle emissioni acustiche rileva passivamente queste onde e le converte in segnali elettrici utilizzando trasduttori piezoelettrici montati sulla superficie, i quali permettono di sfruttare la capacità di alcuni cristalli di produrre segnali elettrici ogni volta che sono sottoposti a sollecitazioni meccaniche.

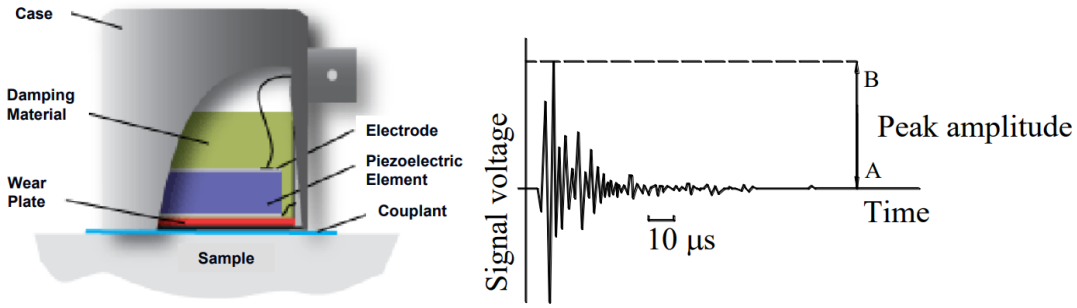

Figura 2.4. Trasduttore piezoelettrico (sinistra) e tipica forma di un segnale ultrasonico (destra)

Il segnale captato da un trasduttore viene preamplificato e, successivamente, viene filtrato per eliminare le frequenze indesiderate associate al rumore, come la vibrazione derivante dalla strumentazione meccanica. Sebbene il nome stesso della tecnica suggerisca un segnale acustico, questo è fuorviante, perché la gamma di frequenze coinvolte è compresa tra 50Khz e 1Mhz, frequenze praticamente ultrasoniche.
I sensori piezoelettrici più comunemente utilizzati nei test AE sono suddivisi in due categorie:

- a larga banda, i quali riescono ad individuare frequenze in vasta gamma risultando, dunque, funzionali nell'analisi dello spettro di frequenza. Sono, però, meno sensibili in ampiezza rispetto ai sensori risonanti;
- risonanti (a banda stretta), i quali sono più sensibili a determinate frequenze, che dipendono dalla frequenza di risonanza dei cristalli presenti all'interno. Inoltre, sono più idonei per la localizzazione della frattura.

L'interpretazione dei segnali di AE può avvenire mediante due distinte metodologie:

- Ring Down Counting, metodo che opera con il conteggio del numero di oscillazioni N_T di un singolo segnale (definito in inglese "hit"), che superano un valore soglia (threshold). Quest'ultimo viene stabilito prima di cominciare il monitoraggio e serve ad eliminare il rumore generato dall'ambiente;

- Counting of Events (in italiano, conteggio degli eventi), metodo secondo cui ogni segnale, ossia l'insieme di tutte le oscillazioni associate ad un microcrack, viene interpretato come singolo evento. Il principio, dunque, alla base di tale procedura è quello di andare a contare il numero di segnali che superano una determinata tensione di soglia: il primo attraversamento è quello che fa partire il conteggio.

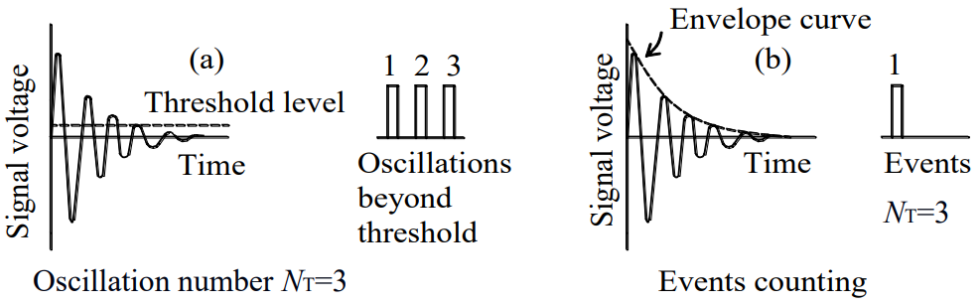

Figura 2.5. Ring Down Counting e Counting of Events

L'intensità dell'evento è misurata attraverso il numero di oscillazioni N_T, il quale aumenta con l'ampiezza del segnale.
Tra i principali sistemi di acquisizione dati utilizzati nel tempo troviamo:

- Apparecchiatura Atel
  Si tratta di un sistema in cui i trasduttori sono impostati su una gamma di frequenze compresa tra 100 kHz e 400 kHz ed è costituito da:

  – un amplificatore
  – un filtro Pbs
  – un misuratore di soglia
  – un contatore di oscillazioni
  – un registratore

  Il livello di soglia è impostato su 100 V ed è amplificato a 100mV, con un guadagno di 60 dB, mentre il limite del conteggio di oscillazioni è fissato a 255 ogni 120 secondi. Il metodo di analisi utilizzato con questa apparecchiatura è il Ring-Down Counting. Come prima approssimazione, il numero di conteggi N può essere confrontato con la quantità di energia rilasciata durante il processo di carico, e si può

presumere che gli incrementi corrispondenti crescano proporzionalmente all'allargamento delle facce della fessura. Questa tecnica prende in considerazione anche altre procedure. Ad esempio, tenendo traccia delle caratteristiche dei trasduttori, e in particolare del suo smorzamento, è possibile considerare tutte le oscillazioni prodotte da un singolo segnale AE come un evento unico e sostituire il Ring-Down Counting con il Counting of events (conteggio degli eventi). L'apparecchiatura Atel non permette la localizzazione della sorgente di emissione acustica.

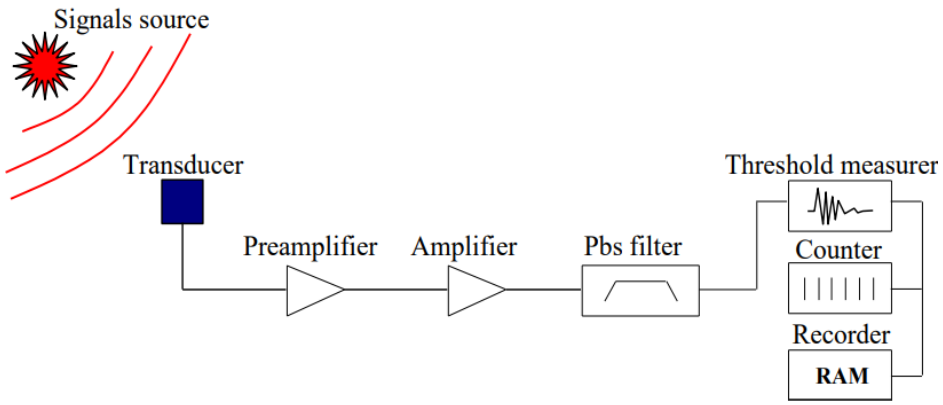

Figura 2.6. Sistema di acquisizione dati Atel

- Apparecchiatura USAM
  Si tratta di un sistema in cui i trasduttori sono impostati su una frequenza compresa tra 50 kHz e 800 kHz e consiste in:

  – 6 sensori, ognuno dei quali è collegato ad un'unità che registra i segnali;
  – Un'unità centrale per la fase di sincronizzazione;
  – Un misuratore di soglia;

  Tramite questa elaborazione si può determinare il grado di danneggiamento della struttura utilizzando un'analisi di tipo parametrico e si possono localizzare le fessure. Non presenta, però, un sistema di analisi automatica dei dati.

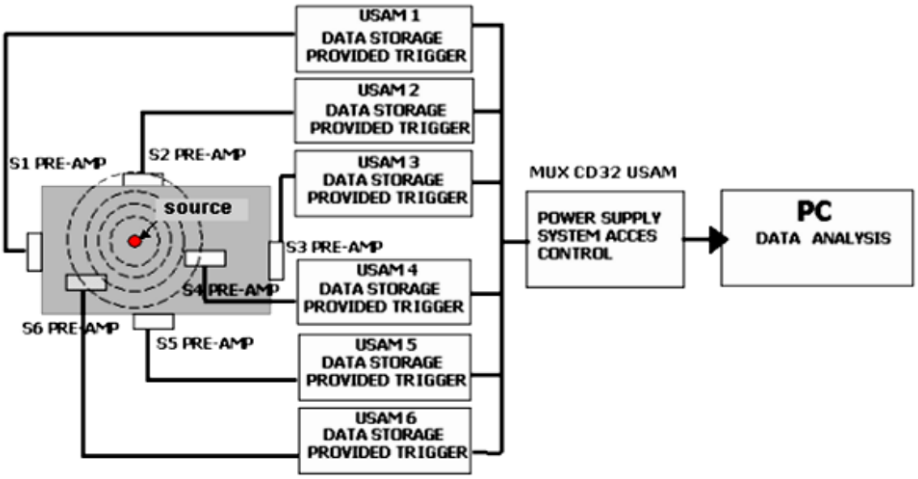

Figura 2.7. Sistema di acquisizione dati USAM

2.3 Analisi parametrica e approccio basato su segnale

La registrazione e l'analisi dei segnali AE possono essere suddivise in due tipologie di approccio: approccio basato su parametri (classico) e approccio basato su segnale (quantitativo).
Nell'analisi basata sui parametri solo alcuni dei parametri del segnale AE vengono acquisiti durante il monitoraggio, ma il segnale stesso non viene registrato. In altre parole, i segnali sono interamente definiti dall'insieme di parametri. Questo approccio consente l'estrazione e l'elaborazione rapida dei dati, poiché riduce al minimo la quantità di dati nel processo di registrazione. Tra i parametri AE di base troviamo:

- Hit, segnale che supera il valore di soglia e avvia l'acquisizione del sistema;
- Ampiezza di picco A (Amplitude), intensità del picco massimo del segnale espressa in Volt o Decibel;
- Tempo di crescita RT (Rise time), intervallo di tempo tra il primo superamento di soglia e l'ampiezza di picco;
- Counts, numero di attraversamenti del valore di soglia;
- Durata (duration), intervallo di tempo tra il primo e l'ultimo attraversamento della soglia;

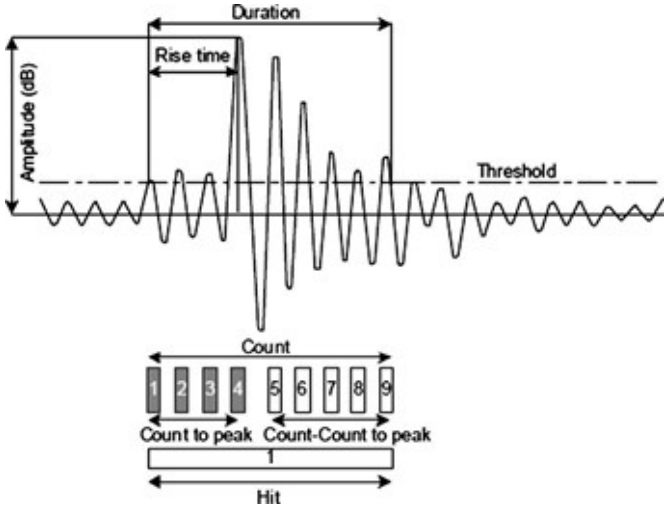

Figura 2.8. Parametri caratteristici di un segnale AE [4]

Tuttavia, con lo sviluppo di una nuova generazione di sensori e la disponibilità di processori computazionali avanzati, è diventata possibile l'acquisizione di forme d'onda grezze multicanale, portando allo sviluppo dell'approccio basato sul segnale. Per acquisire l'onda nella sua interezza è necessario l'oscilloscopio: sebbene esso comporti un costo maggiore, permette di acquisire più informazioni rispetto ad un'analisi parametrica. La caratteristica più importante di questo approccio alla forma d'onda rispetto all'analisi parametrica è la migliore prestazione nella discriminazione del rumore del segnale, offrendo così una migliore interpretazione in alcuni casi. Tale discriminazione è resa possibile dal fatto che è ancora disponibile la forma d'onda originale, la quale non è convertita in una serie di parametri rappresentativi come nel caso dell'analisi parametrica. Pertanto, le forme d'onda possono essere analizzate tramite software, algoritmi o diversi metodi di filtraggio per migliorare il rapporto segnalerumore.
Se da un lato nell'approccio parametrico ci sono possibilità di avere un'elevata registrazione dei dati, oltre all'archiviazione ad alta velocità in modo da facilitarne la visualizzazione rapida, l'implementazione dell'analisi basata sul segnale AE può offrire la possibilità di registrare e memorizzare il maggior numero possibile di segnali lungo la loro forma d'onda in modo che vengano convertiti da segnali analogici a digitali (A/D). Avere segnali digitali consente un'analisi più completa ma dispendiosa in termini di tempo. Inoltre, va notato che l'approccio basato sul segnale AE di solito offre analisi in un ambiente di post-elaborazione, mentre l'analisi parametrica rende possibile il monitoraggio in tempo reale.

2.4 Localizzazione della sorgente di Emissione Acustica

Tra le principali caratteristiche della tecnica AE troviamo la capacità di localizzare il danno. L'identificazione della posizione di origine può consentire un'indagine globale accurata di una struttura e una comprensione preliminare della possibile area danneggiata [12]. Il metodo standard per la localizzazione della sorgente è noto come metodo del tempo di arrivo (Time of Arrival, TOA) [12]. Si tratta di un metodo che è stato ampiamente utilizzato per localizzare la sorgente AE in strutture isotrope in base al tempo di arrivo dei segnali a determinati sensori. La determinazione accurata del primo tempo di arrivo di un segnale è importante per l'accuratezza della posizione della sorgente. Come vedremo analizzando tale metodo, è presente una forte analogia con l'ingegneria sismica in cui la differenza del tempo di arrivo delle onde P ed S può essere usata per determinare la distanza della sorgente. Considerando un mezzo omogeneo e isotropo, la formula che permette di ricavare la distanza della sorgente da una stazione è:

$$d = v_p \cdot v_s \cdot \frac{(t_s - t_p)}{(v_p - v_s)} \tag{2.2}$$

A questo punto è possibile tracciare una sfera avente come centro la stazione e raggio la distanza calcolata. L'intersezione di tre sfere permette la determinazione della posizione dell'ipocentro, seppur grezza poichè il materiale in cui le onde si propagano è stratificato.
Il metodo TOA si basa sull'assunzione di velocità dell'onda isotropica in tutte le direzioni e un percorso di propagazione ininterrotto. Il tempo necessario affinché un'onda di stress si sposti dalla sorgente situata a (x_0, y_0, z_0) a un trasduttore situato a (x_A, y_A, z_A) è calcolato come segue:

$$T_A = \frac{\sqrt{(x_0 - x_A)^2 + (y_0 - y_A)^2 + (z_0 - z_A)^2}}{c} \tag{2.3}$$

dove con c si indica la velocità di trasmissione dell'onda. Tuttavia, questa equazione non può essere utilizzata per determinare la posizione della sorgente, poiché il tempo assoluto di arrivo del segnale ai trasduttori non è noto. Ciò che è noto, invece, sono i tempi relativi di arrivo Δt_a ad ogni trasduttore. L'equazione precedente può essere riscritta introducendo il tempo di arrivo dell'onda ad un trasduttore di riferimento T_R, il quale verrà sottratto da entrambi i membri:

$$\Delta t_a = T_A - T_R = \frac{\sqrt{(x_0 - x_A)^2 + (y_0 - y_A)^2 + (z_0 - z_A)^2}}{c} - T_R \tag{2.4}$$

Partendo da questa equazione, è possibile scrivere un sistema di 4 equazioni in 4 incognite: tempo di arrivo dell'onda al trasduttore di riferimento T_R e la posizione della sorgente (x_0, y_0, z_0). Allo stesso modo, risultano necessari tre trasduttori per determinare la posizione della sorgente nel piano e due nel monodimensionale. Tuttavia, considerando che nel tempo la struttura si danneggia e quindi che c varia durante il

processo di danneggiamento, è conveniente utilizzare nel problema spaziale almeno 5 trasduttori.
Le assunzioni precedentemente elencate per tale metodo risultano non più applicabili nel caso di strutture realistiche, in cui caratteristiche geometriche come fori, bordi irregolari e altre discontinuità strutturali interrompono in modo significativo il percorso e la velocità di propagazione. Inoltre, la distanza di propagazione e il comportamento di dispersione dell'onda rendono difficile determinare con precisione il tempo di arrivo dell'onda. Eventuali errori nella determinazione del tempo di arrivo del segnale comporteranno un'ulteriore perdita di precisione nelle posizioni stimate della sorgente.
Come emerso fino ad ora, la determinazione affidabile ed esatta del tempo di inizio è importante nelle tecniche AE, poiché è la premessa per l'interpretazione dei risultati corrispondenti nella valutazione del danno delle strutture in calcestruzzo. Inoltre, l'enorme quantità di dati acquisiti durante un test rende il rilevamento automatico dell'insorgenza uno stato altamente preferibile.
Negli ultimi decenni sono stati proposti vari metodi di elaborazione del segnale AE per il rilevamento automatico del tempo di insorgenza (Onset time), solitamente scelto come il punto in cui si verifica la prima differenza tra il segnale e il rumore. Tra questi troviamo il metodo proposto dai Professori A. Carpinteri, G. Lacidogna e A. Manuello del Politecnico di Torino, approccio migliorato basato sull'Akaike Information Criterion (AIC) [19]. Il criterio d'informazione di Akaike è un metodo per la valutazione e il confronto tra modelli statistici che fornisce una misura della qualità della stima di un modello statistico tenendo conto sia della bontà di adattamento che della complessità del modello. Nel caso generale, AIC è definito dalla seguente equazione:

$$AIC = -2\ln(L) + 2k \tag{2.5}$$

dove k è il numero di parametri nel modello statistico e L è il valore massimizzato della funzione di verosimiglianza del modello stimato. In genere, si ritiene che un modello con valore AIC minimo sia il più adatto tra i modelli concorrenti. Il minimo globale della funzione AIC definisce il punto di inizio del segnale.
I metodi esistenti utilizzati per il prelievo automatico dell'ora di arrivo AE non possono verificare la validità di ciascun segnale AE rilevato. Questo metodo, invece, consente di determinare un grado di incertezza utile per eliminare i tempi di insorgenza falsi o dubbi e lo fa attraverso due parametri: il grado di certezza DD e velocità apparente V_{ij}.
Il grado di certezza del tempo di insorgenza migliora la precisione della localizzazione del danno AE e può essere stimato dal parametro di certezza DD. Tale parametro, rappresentato dalla derivata seconda della funzione AIC all'istante di insorgenza, è definito come segue:

$$DD = (AIC(k_{min} - \delta k) + AIC(k_{min} + \delta k) - 2AIC(k_{min}))/(\delta k)^2 \tag{2.6}$$

dove k_{min} è il punto che dà il valore minimo a AIC(k) e Δk è definito come il numero di punti corrispondenti a un piccolo intervallo di tempo. Maggiore è il valore DD, più precisa è la determinazione dell'insorgenza. In aggiunta al valore DD, l'altro parametro che può essere considerato per escludere dati dubbi ed errati è la velocità apparente V_{ij}. Questa può essere calcolata tenendo conto dei tempi di insorgenza ottenuti dall'AIC e dalla distanza tra una coppia di sensori i e j che registrano lo stesso evento AE, ovvero:

$$V_{ij} = R_{ij}/|t_i - t_j| \tag{2.7}$$

dove R_{ij} indica la distanza tra i sensori i e j, t_i e t_j sono rispettivamente i tempi di insorgenza calcolati per il segnale AE registrato dai due sensori. Se la velocità apparente è inferiore ad un certo valore, a seconda delle proprietà del materiale dell'elemento monitorato, uno dei due tempi di insorgenza è considerato inaffidabile. Nel caso di strutture in calcestruzzo, il valore di confronto per la velocità apparente è impostato su $4{\cdot}10^3$ m/s, e coincide con la velocità di propagazione delle onde elastiche nel calcestruzzo piano. Quando V_{ij} è maggiore di 4.0 km/s, viene assegnato un punteggio 0 a entrambi i sensori; in caso contrario, viene assegnato loro un punteggio. In conclusione, il livello di affidabilità del segnale può essere valutato dai due parametri sopra elencati (DD e V_{ij}) ed è stato dimostrato che tale metodo è uno strumento affidabile per la determinazione automatica del tempo di insorgenza e utile per la localizzazione della fonte di cricca nelle strutture in calcestruzzo.
Recentemente, sono state studiate e applicate tecniche di Intelligenza Artificiale. In particolare, una ANN ben addestrata potrebbe essere utilizzata per prevedere i risultati senza una buona conoscenza delle funzioni analitiche esplicite. È stato dimostrato che l'applicazione della rete neurale artificiale (ANN) alla posizione della sorgente AE compensa gli effetti dell'anisotropia acustica, delle riflessioni di confine e degli ostacoli nel percorso di propagazione grazie alla sua capacità di gestire problemi complessi. L'addestramento della rete neurale è, in larga misura, legato alla configurazione del sistema di monitoraggio e alle caratteristiche geometriche e fisiche delle strutture bersaglio. Per interpretare i segnali nel modo giusto e stabilire una rete ANN ben addestrata per la posizione della sorgente AE risulta necessaria la stragrande maggioranza del database sperimentale dei test AE. Ottenere tale database tramite esperimenti richiede molto lavoro, tempo e consumo economico. Al fine di ridurre al minimo gli esperimenti richiesti, un modo alternativo consiste nell'utilizzare la simulazione agli elementi finiti (FE) per studiare il meccanismo sottostante alla rilevazione di AE. La maggior parte degli studi esistenti che utilizzano il modello FE per la simulazione della propagazione delle onde si concentra su piastre piatte e geometrie semplici. Facendo riferimento a strutture realistiche, l'analisi FE viene principalmente utilizzata per identificare le regioni di possibili luoghi di danneggiamento che possono quindi essere considerate aree di interesse primario per il monitoraggio strutturale [12].

2.5 Classificazione delle cricche attive

Un altro aspetto molto importante della tecnica AE è la capacità di identificare la modalità di fessurazione a partire da parametri registrati. Determinando il meccanismo alla base dell'evento di fessurazione, la classificazione della modalità di fessurazione svolge un ruolo importante nella comprensione e nella previsione delle probabili modalità di rottura della struttura completa. Inoltre, può aiutare a rilevare lo stato di avanzamento del danno e fornire linee guida per il corretto processo di manutenzione al fine di migliorare la sicurezza e la durabilità strutturale [5].
Per fare ciò si introduce un nuovo parametro, denominato Rise Angle (RA), valutato come rapporto tra il tempo di crescita e l'ampiezza di picco:

$$Rise\ Angle(RA) = \frac{(Rise\ Time(RT))}{(Amplitude(A))} \tag{2.8}$$

Piccoli valori di RA corrispondono ad una fessura di Modo I (apertura della fessura), mentre alti valori di RA corrispondono ad una fessura di Modo II (scorrimento delle facce della fessura).

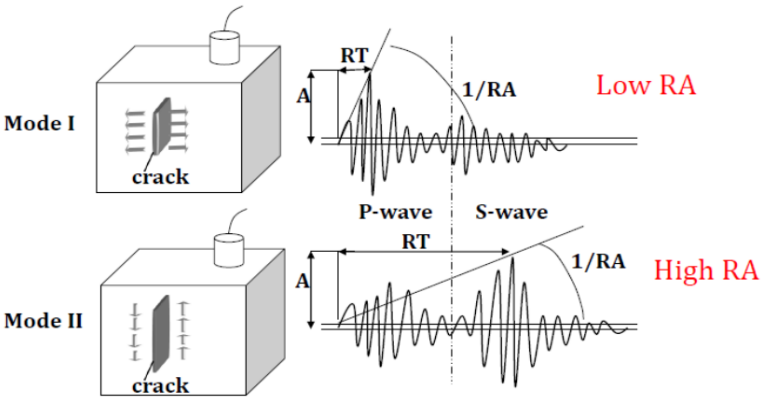

Figura 2.9. Classificazione della modalità di fessurazione con RA [5]

Un altro parametro utilizzato per identificare la modalità di fessurazione è la Frequenza Media (Average Frequency AF), ottenuta come rapporto tra il numero di attraversamenti del valore di soglia e la durata del segnale:

$$Average\ Frequency(AF) = \frac{AE\ Ring\ Down\ Count}{Duration} \tag{2.9}$$

Dalla combinazione delle due grandezze derivate sopra citate è possibile classificare le cricche attive nel materiale in fessure di trazione e taglio.

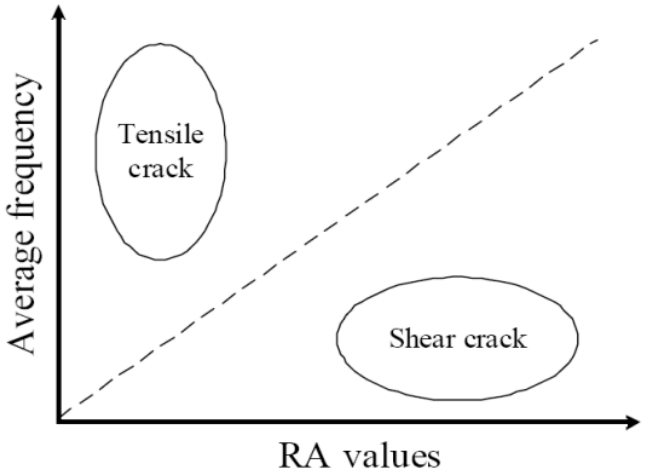

Figura 2.10. Classificazione della modalità di fessurazione con combinazione di RA e AF [5]

Come mostrato nel grafico, le fessure da taglio presentano elevati valori di RA e modesti valori di AF; contrariamente, le fessure da trazione presentano elevati valori di AF e contenuti valori di RA.
In situazioni reali, i cluster corrispondenti a ciascun meccanismo sono sovrapposti e la determinazione del loro confine mediante l'osservazione visiva è inaffidabile. Questo perché per lo stesso insieme di dati possono essere scelti manualmente molti possibili iperpiani, che assumono quindi un carattere soggettivo, e sembrano essere ugualmente validi nel separare i dati. In altre parole, lo stesso iperpiano non può essere replicato da due persone diverse che lavorano con gli stessi dati. Con differenti iperpiani, il rapporto degli eventi nei due cluster, che sono cruciali per la classificazione della modalità di cricca, sarà diverso. Per ovviare a questo problema si è ricorsi all'Intelligenza Artificiale, con il fine di ottenere una separazione oggettiva dei dati AE in cluster. L'algoritmo utilizzato per il raggiungimento di tale obiettivo è il Support Vector Machine (SVM), algoritmo di apprendimento automatico supervisionato basato sull'idea di trovare un iperpiano che divida al meglio un set di dati in due classi.

Capitolo 3

Reti Neurali Artificiali

3.1 Modelli di Reti Neurali

Si richiama in questo capitolo il concetto di Rete Neurale Artificiale. Come visto in precedenza, le Reti Neurali sono un sottoinsieme del Machine Learning e sono l'elemento centrale degli algoritmi di Deep Learning. Il loro nome e la loro struttura sono ispirati al cervello umano, imitando il modo in cui i neuroni biologici si inviano segnali. Esistono varie tipologie di reti neurali, la cui scelta dipende dallo scopo dell'applicazione. Tra le più comuni troviamo:

- il *percettrone*, forma più antica e semplice di rete neurale costituita da un singolo neurone. Analogamente al neurone biologico, il percettrone è costituito da almeno due o più connessioni di input, un nucleo di calcolo e da un'uscita. Grosso limite di questo tipo di architettura è il poter risolvere solo problemi linearmente separabili;
- i *percettroni multilivello* (o MLP, Multi-Layer Perceptron), anche chiamati reti neurali multilivello ad avanzamento (multi-layer feedforward neural network), reti costituite da uno strato di input, uno o più strati nascosti e uno strato di output. Le unità presenti nel livello nascosto sono interamente connesse al livello di input e il livello di output è interamente connesso al livello nascosto. Se tale rete ha più di un livello nascosto prende il nome di rete neurale artificiale profonda (Deep Artificial Neural Network). Queste reti sono caratterizzate dal fatto che le connessioni tra neuroni non formano cicli, ma avanzano in un'unica direzione (feedforward);
- le *Reti Neurali Ricorrenti* (o RNN, Recurrent Neural Network), reti in cui esistono cicli: i valori di uscita di un layer di livello superiore (più vicino all'uscita) vengono utilizzati come ingresso per un layer di livello inferiore (più vicino all'ingresso). Queste interconnessioni tra strati consentono al sistema di creare una memoria;
- le *Reti Neurali Convoluzionali* (o CNN, Convolutional Neural Network), utilizzate principalmente per il riconoscimento delle immagini. Questa tipologia verrà approfondita nel seguito;

3.2 Train, Validation e Test Set

Il primo passo per la creazione di una rete neurale consiste nella raccolta di tutti i dati necessari.
Una delle regole fondamentali nel Machine Learning è quella di non usare lo stesso set di dati per l'addestramento del modello e per la sua valutazione. Risulta, dunque, necessario suddividere il dataset in tre parti ognuno delle quali ha un compito specifico e diverso: training set, validation set e test set [6].

Training set

Il set di addestramento (o training set) è costituito da dati che verranno utilizzati esclusivamente durante la fase di addestramento del modello. In ogni epoca, gli stessi dati di addestramento vengono inviati ripetutamente alla rete neurale e il modello continua ad apprendere le caratteristiche dei dati. Esso imparerà le relazioni tra le variabili x (variabili di input) e y (ovvero il target, l'output). Qui giace il cuore del Learning dei modelli di Intelligenza Artificiale: osservano, studiano, analizzano, e poi cercano di fare delle previsioni su quanto imparato;

Validation set

Il set di convalida (o validation set) è un insieme di dati, separato dal set di addestramento, utilizzato per convalidare le prestazioni del modello durante l'addestramento e ottimizzare gli iperparametri. Il modello vede occasionalmente questi dati, ma non "impara" mai da essi. In altre parole, il modello viene addestrato sul training set e, contemporaneamente, la sua valutazione viene eseguita sul set di validazione dopo ogni epoca. L'idea principale di suddividere il set di dati in un set di validazione è evitare che il modello si adatti eccessivamente ai dati di addestramento, perdendo la capacità di generalizzare e fare previsioni accurate su dati che non ha mai visto prima.

Test set

Il set di test (o test set) è un set separato di dati utilizzato per testare il modello dopo aver completato il training. Partiamo dal presupposto che l'obiettivo dell'addestramento è ottenere una rete neurale che catturi le regolarità presenti nel training set e che, in seguito, sappia fare delle buone previsioni per nuovi dati (ovvero dati non presentati nella fase di addestramento), cioè che abbia capacità di generalizzare. Il set di test rappresenta, dunque, l'insieme di nuovi dati su cui il modello verrà testato. In parole semplici, questo set risponde alla domanda "Come funziona il modello?"

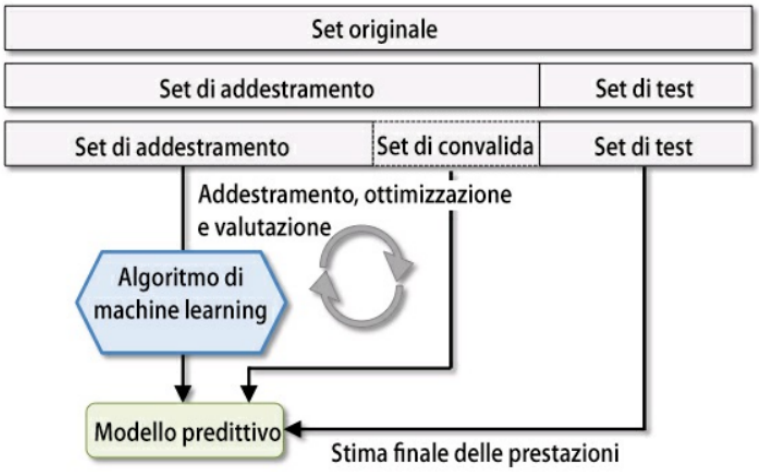

Figura 3.1. Rappresentazione della suddivisione di un dataset [6]

A questo punto sorge spontanea la domanda, come bisogna suddividere i dati di Machine Learning? La risposta è che non esiste una percentuale di suddivisione ottimale, in quanto essa dipende principalmente da due cose: dal numero totale di dati disponibili e, in secondo luogo, dal modello effettivo che si vuole addestrare. L'obiettivo è arrivare a una percentuale divisa che soddisfi i requisiti e le esigenze del modello.
In linea generale, i dati più corposi sono quelli del training set perché l'algoritmo ha bisogno di imparare. Per quanto riguarda il validation set, la percentuale è strettamente connessa al numero di iperparametri da ottimizzare: i modelli con pochissimi iperparametri saranno facili da convalidare e ottimizzare, quindi, probabilmente si potranno ridurre le dimensioni del validation set; in caso contrario, se sono presenti diversi iperparametri il modello richiede un set di convalida più ampio. Solitamente, una buona divisione prevede di utilizzare 70 % dei dati per il training, il 15 % per il validation e il 15 % per il test.

3.3 Iperparametri

Nel campo del machine learning, si hanno due tipi di parametri: quelli che vengono appresi dai dati di addestramento (per esempio i pesi) e i parametri dell'algoritmo di apprendimento, che vengono ottimizzati separatamente [6]. Questi ultimi sono i parametri di ottimizzazione, o iperparametri, di un modello: si tratta di un insieme di variabili esterne al modello che determinano la struttura della rete neurale e il modo in cui viene addestrata. Sostanzialmente, gli iperparametri svolgono un ruolo fondamentale nel decidere se la rete neurale addestrata è applicabile al problema che si sta cercando di risolvere o meno. Tra gli iperparametri troviamo:

Learning Rate

Il Learning Rate (o tasso di apprendimento) è l'iperparametro che controlla la velocità con cui il modello si adatta al problema; in altre parole, esso gestisce lo step di cambiamento dei pesi a ogni iterazione in funzione del gradiente. Si tratta di un valore compreso tra 0 e 1 che viene generalmente indicato con la lettera greca η. Come mostrato in figura 3.2, per η troppo piccolo il tempo di convergenza è grande; quindi, la determinazione del punto di minimo impiega parecchie risorse computazionali; tuttavia, per η grande la convergenza potrebbe fallire, superando il minimo, o arrivando persino a divergere mancando la determinazione della soluzione.

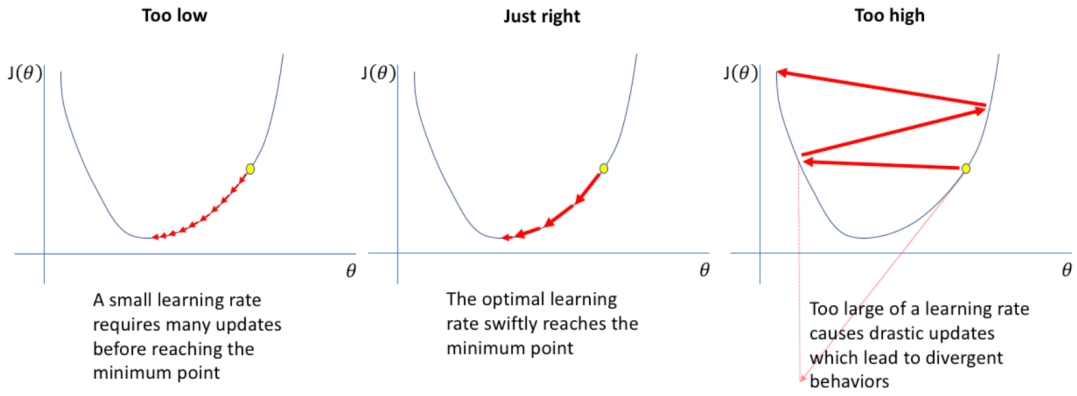

Figura 3.2. Andamento della ricerca del minimo al variare del learning rate

Come con tutti gli iperparametri, è uno scenario di tentativi ed errori. Tuttavia, come vedremo nel seguito, un tasso di apprendimento decrescente (che diminuirà con il progredire della formazione) è preferibile rispetto a un tasso di apprendimento fisso.

Numero di epoche

Indica il numero di volte in cui l'algoritmo di apprendimento passerà attraverso l'intero

set di dati di addestramento. L'epoca rappresenta, dunque, l'unità di misura del tempo di addestramento. A dispetto di quanto si possa pensare, un maggior numero di epoche non porta necessariamente ad un modello più accurato. Questo concetto è chiaramente rappresentato nella figura sottostante:

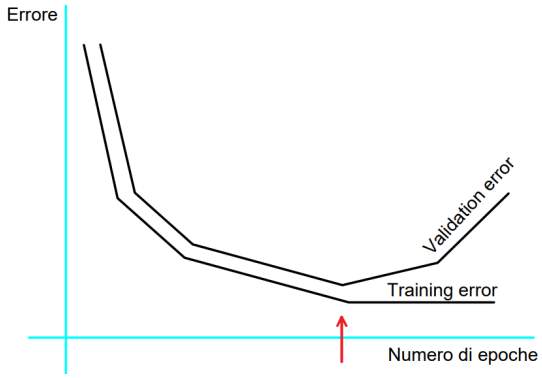

Figura 3.3. Andamento della funzione di costo all'aumentare del numero di epoche

Inizialmente, all'aumentare del numero di epoche, diminuisce sia l'errore sul training set che l'errore sul validation set, ma solo fino a un certo punto. Arrivati al punto indicato in figura dalla freccia, l'errore sul validation set inizia ad aumentare a causa di un fenomeno che prende il nome di overfitting. Quello che succede è che all'aumentare del numero di epoche si può andare in contro ad un eccessivo adattamento ai dati forniti in input senza quindi permettere alla rete di generalizzare. Al contrario, per un basso numero di epoche si verifica il fenomeno di underfitting, ovvero un sotto-allenamento, con conseguenti errori molto elevati. Pertanto, affinché il modello funzioni bene a fronte di nuovi dati, l'addestramento dovrebbe terminare nel punto in cui sia l'errore di addestramento che quello di convalida sono al minimo e il numero di epoche in quel punto è il numero ideale di epoche.

Numero di unità e livelli nascosti

Si tratta di iperparametri che dipendono fortemente dalle dimensioni del problema a cui vengono applicati. All'aumentare della complessità del problema, saranno necessari numeri più elevati di strati nascosti e unità nascoste. Tuttavia, va tenuto presente che un numero maggiore indesiderato di strati e unità nascoste non solo sarà uno spreco di potenza di calcolo, ma comporterà anche un overfitting che ridurrà la precisione della rete neurale. Come principio generale, un metodo per evitare il rischio di eccedere nell'uso di questi iperparametri è di avviare la formazione di una rete neurale molto piccola. Se non si riesce a trovare una soluzione soddisfacente, si fa crescere la rete con l'aggiunta di una ulteriore unita nascosta, fino a quando non si determina la struttura idonea della rete neurale per soddisfare i requisiti del set dei dati.

Batch size

Quando il dataset è di dimensioni notevoli, anche con elaboratori molto performanti non è solitamente possibile elaborare l'intero training set in un colpo solo per limiti imposti dalla memoria e dalla potenza di calcolo. È possibile suddividere i dati di training in sottoinsiemi di dimensione prefissata chiamati batch. Il numero di esempi contenuti in ogni batch, e quindi il numero di dati (campioni) che verranno addestrati dalla rete neurale in una volta sola, è chiamato batch size. Questo diminuisce notevolmente l'utilizzo della memoria centrale.

3.4 Addestramento di una Rete Neurale

Da un punto di vista "fisico", una Rete Neurale è composta da unità chiamate neuroni, arrangiati in strati (layers) e collegati tra di loro tramite connessioni pesate, con due caratteristiche fondamentali:

- la "conoscenza" è acquisita dall'ambiente esterno attraverso un processo di apprendimento;
- la "conoscenza" è immagazzinata nei parametri della rete e, in particolare, nei pesi associati alle connessioni.

In altre parole, la rete neurale non viene programmata in modo diretto ma addestrata esplicitamente attraverso un algoritmo di apprendimento, per risolvere un dato compito, con un processo che porta all'apprendimento tramite l'esperienza.
L'algoritmo di apprendimento è uno degli elementi più significativi tra i fattori (numero di strati, numero di neuroni, etc.) che concorrono a definire la configurazione specifica di una rete neurale e che, quindi, condizionano e determinano la capacità stessa della rete di fornire risposte corrette allo specifico problema. È possibile individuare due paradigmi fondamentali:
-***Apprendimento supervisionato*** (Supervised Learning): quando il set utilizzato per l'addestramento della rete contiene sia i dati di ingresso che i relativi dati di uscita reali;
-***Apprendimento non supervisionato*** (Unsupervised Learning): quando i valori del modello della rete neurale artificiale vengono modificati solo in funzione dei dati di ingresso.
Un'altra distinzione importante, relativa alle metodologie di apprendimento, riguarda le modalità con cui viene acquisito o utilizzato l'insieme di campioni di training durante l'apprendimento. Da questo punto di vista si può distinguere:
-***Apprendimento batch o fuori linea***: si suppone che tutto l'insieme di addestramento sia disponibile prima che l'addestramento abbia inizio. Le modifiche ai pesi vengono apportate solo dopo che alla rete sono stati presentati tutti i dati nel set di training. In altre parole, si valuta l'errore della rete sull'intero insieme degli esempi prima di aggiornare i pesi; l'idea è quella di fare poche modifiche ma sostanziali;
-***Apprendimento on-line***: in questo caso gli esempi del training set vengono acquisiti (o utilizzati) in modo incrementale durante il processo di addestramento. Le modifiche avvengono subito dopo aver preso in considerazione ogni punto del set di dati; si procede quindi con tante piccole modifiche. Questo offre numerosi vantaggi:

- spesso è molto più veloce, soprattutto quando il training set è ridondante (contiene molti punti di dati simili);
- può essere utilizzato quando non vi è un training set prefissato;
- il rumore del gradiente può aiutare a uscire dai minimi locali (che sono un problema per l'algoritmo della discesa del gradiente nei modelli non lineari).

-***Apprendimento a mini-batch***: compromesso tra i due metodi di apprendimento sopra citati è l'uso di algoritmi di apprendimento a mini-batch, per i quali i pesi sono aggiornati dopo ogni n punti di dati, dove n è maggiore di 1, ma più piccolo della dimensione massima del set dei dati.
Da un punto di vista pratico, l'addestramento di una rete neurale consiste nel determinare i valori dei pesi delle connessioni tra tutti i nodi e gli eventuali bias, in modo tale da mappare il più accuratamente possibile le relazioni che sussistono tra input e output. Per capire come questo avviene, risulta necessario introdurre un nuovo parametro: la Funzione Costo.

3.4.1 Funzione Costo

Al fine di precisare ciò che si intende per "raggiungimento di un buon indice di previsione", si definisce la funzione costo (cost function, o funzione d'errore). Questa funzione fornisce una misura oggettiva dell'errore, inteso come distanza che intercorre tra gli output desiderati e i corrispondenti output che il modello calcola. Tra le funzioni di costo più frequentemente utilizzate si trova l'errore quadratico medio (MSE, mean squared error), che consiste nella media aritmetica delle differenze degli errori di previsione al quadrato:

$$MSE = \frac{\sum_{i=1}^{n}(\hat{y}_i - t_i)^2}{n} \tag{3.1}$$

In altre parole, è la somma per tutti i punti i nel set di dati del quadrato della differenza tra il valore effettivo t_i (target) e il valore del modello di previsione y_i, calcolato con valore di ingresso x_i. Il suo frequente utilizzo deriva dal fatto che si tratta di una funzione convessa (convex function), la quale presenta due principali vantaggi:

- Ha un minimo globale, senza alcun minimo locale;
- La random initialization non compromette la convergenza.

Spesso, è utilizzato nella forma "rooted" (in questo caso si parla di RMSE), in quanto ha il vantaggio di riportare la media sullo stesso ordine di grandezza dei valori originali:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - t_i)^2}{n}} \tag{3.2}$$

A questo punto, si può quindi riformulare l'obiettivo di individuare il migliore modello come trovare i valori dei parametri che minimizzano tale funzione. Per il raggiungimento di questo scopo, entrano in gioco gli algoritmi di ottimizzazione (optimization algorithm).

3.4.2 Algoritmi di ottimizzazione

Il concetto di ottimizzazione gioca un ruolo chiave quando si parla di machine learning e, in particolare, di deep learning. Lo scopo principale degli algoritmi di deep learning è quello di costruire un modello di ottimizzazione che, tramite un processo iterativo, minimizzi una funzione obiettivo, che in questo caso risulta essere la cost function. In altre parole, attraverso iterazioni multiple questi algoritmi consentono l'individuazione dei weights che minimizzano la funzione costo.
I metodi di ottimizzazione possono essere suddivisi in due categorie: metodi del primo ordine, basati sul concetto di discesa del gradiente, e metodi di ottimizzazione del secondo ordine o di ordine superiore, fra i quali il metodo di Newton ne è un tipico esempio. Di seguito si riportano i più popolari algoritmi di ottimizzazione del primo ordine.

Discesa del Gradiente

Il Gradient Descent (o discesa del gradiente) è il primo e più comune metodo di ottimizzazione. Per comprendere tale algoritmo in maniera intuitiva, si immagini di essere su una montagna e di voler raggiungere il punto più basso, ovvero la valle. Per iniziare ci si guarda intorno con l'obiettivo di trovare la pendenza migliore per scendere. Una volta trovata, si fa un passo avanti di una magnitudine α. Questo è esattamente il principio della Discesa del Gradiente: l'idea è quella di trovare il minimo di una funzione matematica, ma senza risolverla in modo "tradizionale". Si imposterà, invece, un algoritmo iterativo che si sposterà passo dopo passo per avvicinarsi alla soluzione e quindi trovare il minimo.
Da un punto di vista matematico, il gradiente di una funzione è definito come il vettore che ha per componenti cartesiane le derivate parziali della funzione. Esso rappresenta, quindi, la direzione di massimo incremento di una funzione di n variabili $f(x_1, x_2, ..., x_n)$. Poiché l'obiettivo è quello di raggiungere il minimo di una funzione, è nostro interesse ottenere un gradiente negativo. Quindi, si aggiorneranno i pesi nella direzione negativa del gradiente. Per stabilire di quanto deve essere aggiornato il valore del peso, entra in gioco il learning rate, iperparametro che (come anticipato in precedenza) gestisce lo step di cambiamento dei pesi a ogni iterazione in funzione del gradiente. Questa tecnica opera secondo i seguenti passaggi:

1. Inizializzazione dei pesi della rete a valori casuali (random initialization)

2. Propagazione dei dati iniziali attraverso la rete fino ad ottenere un output

3. Comparazione dei risultati ottenuti con quelli supervisionati e valutazione della funzione costo

4. Calcolo del gradiente della funzione di costo rispetto ai pesi

5. Aggiornamento dei pesi in modo tale da muoversi nella direzione negativa del gradiente per minimizzare la "perdita"

6. Ripetere i passaggi 4 e 5 fino a quando il valore del gradiente si avvicina a zero.

L'aggiornamento del peso avviene mediante la seguente formula:

$$w_{t+1} = w_t - \eta \frac{\partial E}{\partial w_t} \tag{3.3}$$

L'algoritmo termina una volta che si è sufficientemente vicini al minimizzare la funzione di costo, cioè dove E = 0. Si dice allora che l'algoritmo converge. In sintesi: A

	Caso generale
Formazione dei dati	(x, t)
Parametro del modello	w
Modello	$y = g(w, x)$
Funzione Errore	$E(y, t)$
Gradiente rispetto w_{ij}	$\frac{\partial E}{\partial w_{ij}}$
Regola aggiornamento dei Pesi	$\Delta w_{ij} = -\mu \frac{\partial E}{\partial w_{ij}}$

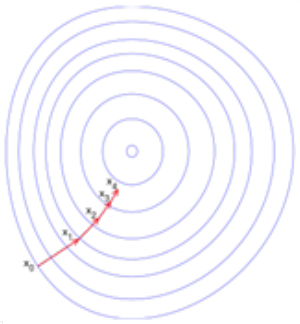

Figura 3.4. Sintesi dell'algoritmo di Gradient Descent e illustrazione grafica del metodo

seconda della quantità di dati utilizzati, sono definite diverse varianti della discesa del gradiente. Nel seguito vengono discussi tre tipi popolari:

Discesa del gradiente batch

Chiamata anche Vanilla Gradient Descent, è il primo tipo di base di discesa del gradiente. Calcola l'errore per ogni esempio all'interno del set di dati di addestramento, ma solo dopo che tutti gli esempi di addestramento sono stati valutati il modello viene aggiornato. Tale calcolo è costoso su un set di dati molto grande; inoltre, deve memorizzare pesi intermedi in memoria ad ogni iterazione e non consente di effettuare gli aggiornamenti online di α.
Insomma, la discesa del gradiente batch risulta essere un algoritmo lento. Tuttavia, l'utilizzo dell'intero set di dati è davvero utile per arrivare ai minimi in modo meno rumoroso e meno casuale.

Discesa del gradiente stocastica

In presenza di dataset molto estesi, un'alternativa comune all'algoritmo a Discesa del Gradiente batch è quello di Discesa del Gradiente Stocastica (Stochastic Gradient Descent, SGD), chiamato anche discesa del gradiente iterativa o online.

La parola "stocastico" indica un sistema o un processo collegato a una probabilità casuale. Infatti, in questo caso l'idea è quella di calcolare il gradiente non più mediante tutto il training set, ma utilizzando un singolo campione selezionato in modo casuale ad ogni iterazione. Sebbene tale algoritmo possa essere considerato un'approssimazione della discesa del gradiente, in genere raggiunge la convergenza molto più velocemente, poiché i pesi vengono aggiornati più frequentemente. Poiché ciascun gradiente viene calcolato sulla base di un singolo esempio di addestramento, la superficie di errore è più rumorosa rispetto alla Discesa del Gradiente batch. Tuttavia, questo porta al vantaggio di essere in grado di sfuggire con maggiore facilità agli avvallamenti rappresentati dai minimi locali.
Nelle implementazioni della discesa del gradiente stocastica, il tasso fisso di apprendimento viene frequentemente sostituito da un tasso di apprendimento adattativo che si riduce nel corso del tempo. Si noti che la discesa del gradiente stocastica non raggiunge il minimo globale, ma una zona che gli si avvicina molto. Utilizzando un tasso di apprendimento adattativo, possiamo ottenere un ulteriore affinamento, per individuare un minimo globale migliore.
Un altro vantaggio della Discesa del Gradiente Stocastica è che possiamo utilizzarla per l'apprendimento online.

Discesa del gradiente mini-batch

L'utilizzo di una selezione casuale di singoli campioni dal training set ha lo svantaggio di determinare una oscillazione della direzione del gradiente e un procedere in modo cieco del processo di ricerca della soluzione. Un compromesso fra Discesa del Gradiente Batch e Discesa del Gradiente Stocastica è la Mini-Batch Gradient Descent. Questa modalità impiega, ad ogni iterazione, un sottoinsieme di k degli n campioni del training set (con 1¡k¡n) e con questi calcola il gradiente. In questo modo si raggiunge il duplice obiettivo di ridurre la varianza del gradiente e rendere più stabile la convergenza.

Metodo del Momento

Nonostante il metodo SGD sia molto popolare ed ampiamente utilizzato, esistono condizioni tali per cui può diventare molto lento: in presenza di curvature elevate, gradienti piccoli ma coerenti o gradienti rumorosi. Ciò è dovuto alla regola di aggiornamento dell'algoritmo, che dipende solo dai gradienti a ciascuna iterazione. Per mitigare questi problemi e accelerare l'apprendimento delle reti neurali, nel 1964 Polyak introdusse il Metodo del Momento. Si tratta di un'estensione dell'algoritmo di Discesa del Gradiente, spesso indicato come discesa del gradiente con quantità di moto. Per comprendere tale algoritmo in maniera intuitiva, si immagini una palla che rotola lungo un dolce pendio su una superficie liscia: inizierà lentamente, ma prenderà rapidamente slancio fino a raggiungere la velocità terminale. Questa è l'idea molto semplice dietro a tale

algoritmo. Al contrario, la normale discesa del gradiente farà semplicemente piccoli passi regolari lungo il pendio, quindi l'algoritmo impiegherà molto più tempo per raggiungere il fondo.
L'innovazione di tale metodo risiede nel fatto che, nella regola di aggiornamento dei pesi, non si considerano più solamente le informazioni relative al gradiente della funzione costo al passo corrente, ma si tiene conto anche di come il gradiente sta evolvendo nel tempo. L'idea è quella di calcolare, ad ogni iterazione, una media mobile esponenziale dei gradienti attuali e storici (cioè fino al tempo t) ed utilizzare questo valore come direzione da seguire nell'aggiornamento dei pesi. Questo può essere chiaramente osservato nell'equazione della regola di aggiornamento, cioè:

$$m_{t+1} = \beta \cdot m_t - \eta \cdot \frac{\partial E}{\partial w_t} \tag{3.4}$$

$$w_{t+1} = w_t + m_{t+1} \tag{3.5}$$

dove:

- β è il momentum learning, iperparametro che determina la velocità di decadimento dei contributi dei gradienti precedentemente accumulati e il cui valore è compreso tra 0 e 1 (tipicamente viene imposto circa a 0.9);
- m è il momentum vector che, per come è stato pensato, gioca il ruolo di una velocità;
- t indica il passo corrente o epoch.

Se β è molto più grande di η, i gradienti precedenti accumulati saranno dominanti nella regola di aggiornamento, quindi il gradiente all'iterazione non cambierà rapidamente la direzione corrente. Questo è positivo quando il gradiente è rumoroso. D'altra parte, se β è molto più piccolo di η, i gradienti precedenti accumulati possono agire come un fattore di smussamento per il gradiente.

AdaGrad e RMSProp

Con l'introduzione del tasso di apprendimento si è sottolineato come una sua opportuna regolazione potesse avere una forte influenza sugli effetti del metodo di Discesa del Gradiente. A tal riguardo sono stati proposti diversi meccanismi adattativi per una regolazione automatica del learning rate. Tra questi rientra l'Adaptive Gradient, o in breve AdaGrad, un'estensione dell'algoritmo di ottimizzazione della Discesa del Gradiente introdotta nel 2011 da John Duchi et al. nell'articolo: "*Adaptive Subgradient Methods for Online Learning and Stochastic Optimization*".
Il cambiamento introdotto da AdaGrad risiede nel fatto che, durante il processo di aggiornamento dei pesi, il learning rate non sarà più costante ma sarà continuamente ricalcolato utilizzando i valori storici del gradiente accumulati fino alla corrente iterazione.

L'algoritmo si basa sulle seguenti relazioni:

$$g_t = \nabla_{w_t} E(w_t) \tag{3.6}$$

$$v_t = \sqrt{\sum_{i=1}^{t} (g_i)^2 + \epsilon} \tag{3.7}$$

$$w_{t+1} = w_t - \eta \frac{g_t}{v_t} \tag{3.8}$$

dove ϵ è un termine di smoothing, tipicamente fissato pari a 10^{-10}, introdotto per impedire la divisione per zero nell'implementazione.
Uno dei principali vantaggi di Adagrad è che elimina la necessità di regolare manualmente il tasso di apprendimento. Il principale punto debole è l'accumulo dei gradienti quadratici nel denominatore: poiché ogni termine aggiunto è positivo, la somma accumulata continua a crescere durante l'allenamento. Ciò fa sì che il tasso di apprendimento si riduca e, nei casi di training time molto lunghi, diventi infinitamente piccolo. Di conseguenza, i pesi tenderanno a valori stazionari non corretti.
In altre parole, il tasso di apprendimento viene ridotto così tanto che l'algoritmo finisce per fermarsi completamente prima di raggiungere l'optimum globale.
Tale problema è stato affrontato e risolto da Geoffrey Hinton e Tijmen Tieleman con il metodo Root Mean Squared Propagation, o in breve RMSProp, proposto nel 2012. Si tratta di un'estensione di AdaGrad in cui si utilizza una media mobile delle derivate parziali invece della somma, consentendo alla ricerca di dimenticare i primi valori delle derivate parziali. In breve, tale algoritmo accumula solamente i gradienti delle iterazioni più recenti. Ciò si ottiene aggiungendo un nuovo iperparametro che chiameremo β, il quale agisce come quantità di moto per le derivate parziali. L'algoritmo diventa dunque:

$$v_t = \beta \cdot v_{t-1} + (1 - \beta) \cdot g_t^2 \tag{3.9}$$

Il tasso di decadimento β è tipicamente impostato a 0,9. Si tratta ancora una volta di un nuovo iperparametro, ma questo valore predefinito spesso funziona bene; quindi, potrebbe non essere necessario ottimizzarlo affatto.

Adam

Adam (Adaptive moment estimation) è un metodo di apprendimento adattivo presentato nel 2015 da Diederik P. Kingma e Jimmy Ba nel loro articolo intitolato "*Adam: A Method for Stochastic Optimization*" [20]. Come sottolineato nell'articolo, si tratta di un metodo progettato per integrare in modo efficace i vantaggi di due algoritmi precedentemente introdotti: AdaGrad, che funziona bene con gradienti sparsi, e RMSProp, che funziona bene in impostazioni on-line e non stazionarie.
Per adattare la velocità di apprendimento per ogni peso della rete neurale, Adam utilizza medie mobili esponenziali del gradiente (m_t) e del gradiente quadrato (v_t). Le

medie mobili stesse sono stime del primo momento (media) e del secondo momento grezzo (varianza non centrata) del gradiente:

$$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t \tag{3.10}$$

$$v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2 \tag{3.11}$$

dove gli iperparametri $\beta_1, \beta_2 \in [0, 1)$ ne controllano i tassi di decadimento esponenziale. Tuttavia, queste medie mobili sono inizializzate come vettori di 0, portando a stime del momento che sono polarizzate verso lo zero, specialmente durante i tempi iniziali e quando i tassi di decadimento sono piccoli (per esempio quando i β_s sono vicini a 1). Per contrastare questi bias di inizializzazione, si introducono stime corrette del primo e del secondo momento:

$$\hat{m}_t = \frac{m_t}{(1 - \beta_1^t)} \tag{3.12}$$

$$\hat{v}_t = \frac{v_t}{(1 - \beta_2^t)} \tag{3.13}$$

A questo punto è possibile eseguire l'aggiornamento del peso come segue:

$$w_t = w_{t-1} - \eta \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t + \epsilon}} \tag{3.14}$$

In questo caso, buoni valori predefiniti sono α=0.001, β_1=0.9, β_2=0.999 e $\epsilon = 10^{-8}$.
Gli autori di tale algoritmo elencano nel loro articolo gli interessanti vantaggi dell'utilizzo di Adam su problemi di ottimizzazione: semplice da implementare, computazionalmente efficiente, pochi requisiti di memoria, invariante al ridimensionamento diagonale dei gradienti, adatto per problemi di grandi dimensioni in termini di dati e/o parametri, adatto per obiettivi non stazionari, adatto per problemi con pendenze molto rumorose e/o sparse; infine, gli iperparametri hanno un'interpretazione intuitiva e in genere richiedono poca messa a punto.

3.4.3 Algoritmo di Error back propagation

Fin qui si sono analizzati alcuni dei più famosi algoritmi di ottimizzazione del primo ordine e si è messo in luce come questi si basino sul concetto di discesa del gradiente. Nel tentativo di applicare tali metodi a reti neurali complesse a più livelli (reti multilayer), però, ci si imbatte in una ulteriore difficoltà: sui livelli intermedi della rete (hidden layers) non si hanno a disposizione tutti i valori di "riferimento" attesi (uscite desiderate). Di conseguenza, non risulta possibile valutare la funzione costo necessaria per l'aggiornamento dei pesi. Dunque, sorge spontanea la domanda: in che modo si potrebbe dire alle unità nascoste come aggiornare i pesi delle loro connessioni? Ed è qui che entra in gioco l'algoritmo di Retropropagazione dell'errore.
L'Error Back-Propagation (backpropagation, BP) è un algoritmo ampiamente utilizzato nella formazione di reti neurali per l'apprendimento supervisionato. In linea di principio

la Retropropagazione dell'errore fornisce un metodo per formare le reti neurali con un qualsiasi numero di unità nascoste disposte in un qualsiasi numero di strati. Tali reti devono rispettare il vincolo di essere di tipo feedforward, cioè le connessioni devono andare da unita "precedenti", vicino all'ingresso, a unità "successive", vicine all'uscita. In altre parole, il loro modello di collegamento non deve contenere cicli. La BP prevede due fasi, una fase in avanti e una fase indietro:

- Fase in avanti (forward): l'input fornito alla rete viene propagato ai livelli successivi fino a determinare l'uscita. A questo punto si calcola l'errore E(w);
- Fase indietro (backward): l'errore E(w) viene propagato indietro nella rete, aggiustando progressivamente i pesi.

Si tratta, dunque, di un algoritmo numerico per il calcolo del gradiente di reti feedforward, basato sulla chain rule, tale per cui sappiamo che per calcolare la derivata di una funzione composta è possibile suddividere il calcolo in più step.
Prima di addentrarci nella formulazione matematica del calcolo del gradiente attraverso tale algoritmo, si definiscono le seguenti quantità facendo riferimento alla figura sottostante:

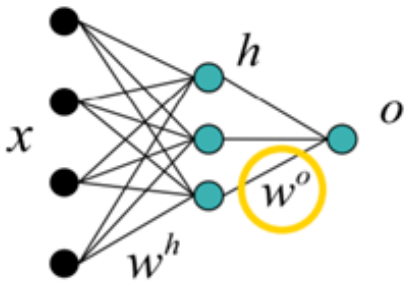

Figura 3.5. Esempio di rete neurale multilayer

$$z^o = \sum_{j=0}^{n} w_j^o h_j \qquad o = \sigma(z^o) \tag{3.15}$$

$$z_j^h = \sum_{i=0}^{m} w_{ij}^h x_j \qquad h_j = \sigma(z_j^h) \tag{3.16}$$

dove con x indichiamo le variabili di input, con o indichiamo l'output e con h indichiamo gli strati nascosti (hidden layers). Le formule a destra, invece, rappresentano l'output fornito ai vari livelli dalla funzione di attivazione. Poiché l'algoritmo di apprendimento BP si basa sul calcolo del gradiente, bisogna assicurarsi che tale funzione sia continua e differenziabile. Pertanto, la funzione di attivazione comunemente usata è la sigmoide, la quale ricalca la funzione soglia con la differenza che il passaggio da 0 a +1 è più graduale (con un andamento a forma di s appunto) rendendola differenziabile.

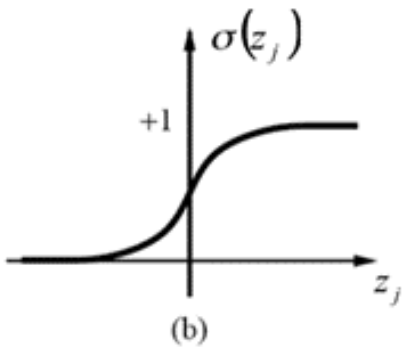

Figura 3.6. Funzione di attivazione sigmoide

$$\sigma(x) = \frac{1}{1+e^{-x}} \qquad \frac{\sigma(x)}{dx} = \sigma\prime(x) = \sigma \cdot (1-\sigma(x)) \tag{3.17}$$

A questo punto è possibile calcolare gli errori associati ai neuroni. Per fare ciò, distinguiamo i due casi possibili:

- il neurone appartiene allo strato d'uscita: in questo caso, la formula per il calcolo del gradiente risulterà essere:

$$\frac{\partial E}{\partial w_j^o} = \frac{\partial E}{\partial o} \cdot \frac{\partial o}{\partial z^o} \cdot \frac{\partial z^o}{\partial w_j^o} \tag{3.18}$$

- il neurone è un neurone nascosto: si deve propagare indietro l'errore dei nodi di uscita (da cui il nome dell'algoritmo). Anche in questo caso, utilizzando il metodo della chain rule, si può esprimere l'errore di una unita nascosta in termini dei suoi nodi successivi:

$$\frac{\partial E}{\partial w_{ij}^h} = \frac{\partial E}{\partial o} \cdot \frac{\partial o}{\partial z^o} \cdot \frac{\partial z^o}{\partial h_j} \cdot \frac{\partial h_j}{\partial z_j^h} \cdot \frac{\partial z_j^h}{\partial w_{ij}^h} \tag{3.19}$$

È bene notare che il termine Error Back-Propagation è spesso usato vagamente per riferirsi all'intero algoritmo di apprendimento. In realtà, esso si riferisce rigorosamente solo all'algoritmo per il calcolo del gradiente, non al modo in cui il gradiente viene utilizzato (che invece è stabilito dal metodo di discesa del gradiente).
In conclusione, si è dimostrato come l'algoritmo della Backpropagation insieme al Gradient Descent, ci permetta di ottimizzare i pesi della rete e di realizzare un processo di apprendimento. Al termine di quest'ultimo, i pesi della rete saranno stati ottimizzati in modo da ridurre l'errore della funzione di costo.

3.5 Reti Neurali Convoluzionali

Le Reti Neurali Convoluzionali, indicate in letteratura con gli acronimi CNN (Convolutional Neural Network) o ConvNet, sono uno degli algoritmi di Deep Learning più utilizzati oggi nel campo della visione artificiale. Esse hanno acquisito popolarità all'interno della Computer Vision grazie alle loro eccezionali prestazioni nella classificazione delle immagini (*Image Classification*) e nell'identificazione di oggetti all'interno di un'immagine (*Object Detection*).
Ci si potrebbe chiedere da dove nasce la necessità di introdurre algoritmi tutto sommato complessi come le convoluzioni per l'analisi di un'immagine: non si potrebbe raggiungere lo stesso scopo utilizzando una rete neurale fully connected (ovvero una rete in cui ogni nodo è collegato con tutti i nodi dei layer precedente e successivo)? La risposta è semplice: benché questa funzioni bene per le piccole immagini, non vale lo stesso per le immagini più grandi a causa dell'elevato numero di parametri che richiede. Il problema è proprio la caratteristica di connessione completa dei neuroni: si supponga, ad esempio, di avere immagini da 200x200x3, dove il terzo valore rappresenta la profondità RGB. In questo caso, un neurone dovrebbe avere un numero di parametri pari a 200·200·3=120˙000. Di conseguenza, l'intera rete avrebbe un numero di parametri enorme rendendola molto difficile da addestrare, se non addirittura impossibile.
Come si vedrà in seguito, le ConvNet risolvono questo problema utilizzando livelli parzialmente connessi e condivisione dei pesi.

3.5.1 Storia delle CNN

L'ispirazione per le CNN arriva ancora una volta dal mondo biologico, in particolare dallo studio della corteccia visiva del cervello. A partire dalla fine degli anni '50 e per tutto il decennio successivo, David Hubel e Torsten Wiesel eseguirono una serie di esperimenti, prima sul gatto e in seguito sulla scimmia, dando spunti cruciali sulle strutture neurali coinvolte nella visione. Il loro lavoro mostrò come la corteccia visiva contenesse neuroni che individualmente rispondono a piccole regioni del campo visivo. In altre parole, essi notarono come ad ogni neurone appartenesse un campo recettivo, definito come la regione dell'area visiva in cui deve cadere lo stimolo per eccitare o inibire il neurone interessato. Inoltre, identificarono due tipi di cellule visive nel cervello: le cellule semplici, che rispondono principalmente a bordi e reticoli orientati, e le cellule complesse, che ricevono input da un numero di cellule semplici e, quindi, hanno campi recettivi più ampi. In seguito, questi studi ispirarono il Neocognitron, rete neurale artificiale gerarchica e multistrato proposta da Kunihiko Fukushima nel 1980. Un'altra pietra miliare fu LeNet-5, un'architettura di rete neurale convoluzionale ampiamente impiegata nel riconoscimento delle cifre. Questa architettura possiede alcuni elementi già noti, come livelli fully connected e funzioni di attivazione sigmoide, ma introduce anche due nuovi elementi costitutivi: livelli convoluzionali e livelli di pooling. [7]

3.5.2 Architettura delle CNN

La *Convolutional Neural Network* è un tipo di rete neurale feed-forward ispirata all'organizzazione della corteccia visiva del cervello umano.
Una CNN è composta da una gerarchia di livelli: il livello di input, direttamente collegato ai pixel dell'immagine, un alternarsi di strati convoluzionali e di pooling fino ad arrivare all'ultimo strato completamente connesso che produce l'uscita della rete. Ad ogni livello, la CNN aumenta nella sua complessità identificando porzioni maggiori dell'immagine, poiché i livelli successivi possono vedere i pixel all'interno dei campi recettivi dei livelli precedenti. In altre parole, il campo recettivo dei neuroni aumenta muovendosi verso l'alto nella gerarchia.
In particolare, gli strati in prossimità dell'input estraggono dall'immagine features spaziali, come ad esempio orientamento dei contorni, presenza di angoli, informazioni sul colore, etc.. Avvicinandosi allo strato di output verranno estratte features sempre più generali e complesse, fino ad includere quelle che definiscono l'oggetto in modo univoco.

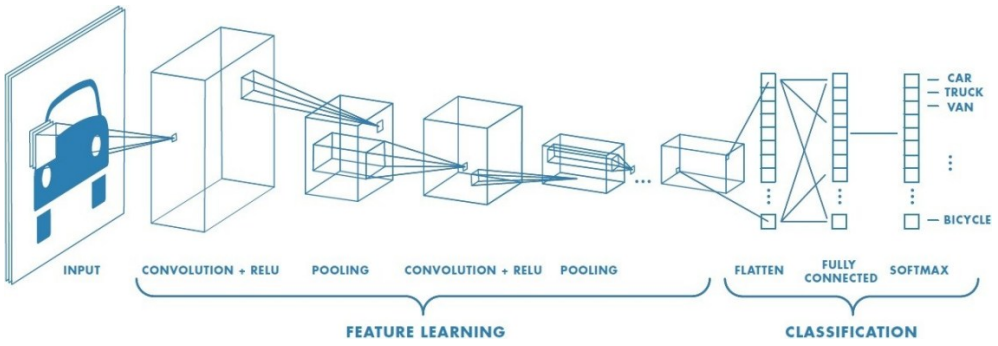

Figura 3.7. Esempio di una rete neurale convoluzionale con numerosi layer convoluzionali

Come emerge in *Figura 3.7*, una rete CNN è suddivisa in due processi:

- il primo (*feature learning*) racchiude i livelli iniziali dove vengono estratte le feature dall'immagine utilizzando i processi di convoluzione e pooling;
- il secondo processo (*classification*) è costituito da una rete neurale tradizionale completamente connessa per classificare al meglio l'immagine in base alle feature ricevute in input.

Analogamente a una rete neurale tradizionale, una CNN possiede neuroni con pesi e bias. Il modello apprende questi valori durante l'addestramento e li aggiorna costantemente con ogni nuovo esempio di addestramento. Tuttavia, nel caso delle CNN, i valori dei pesi e dei bias sono gli stessi per tutti i neuroni nascosti in un determinato layer. Ciò significa che tutti i neuroni nascosti rilevano la stessa feature in diverse aree dell'immagine. Ciò rende la rete tollerante alla traslazione di oggetti in un'immagine. Di seguito si andranno ad analizzare i tre tipi di strati che si hanno a disposizione per

la realizzazione di una Rete Neurale Convoluzionale:

Strato convoluzionale o Convolutional Layer

Ciò che differenzia le CNN rispetto alle classiche reti feed forward è la presenza dei cosiddetti livelli convolutivi, i quali conferiscono il nome alla rete. Il loro compito è quello di eseguire un'operazione chiamata "convoluzione". Matematicamente parlando, la parola convoluzione significa "far scorrere" una funzione sopra un'altra, di fatto "mescolandole" assieme: il risultato sarà una funzione che rappresenta il prodotto delle due funzioni. Questa è una definizione rigorosamente matematica del processo, ma in che modo questa operazione può interessare le immagini?
Nel caso dell'analisi di immagini, le operazioni di convoluzione permettono di applicarle filtri specifici per estrarre un set di caratteristiche (features). La moltiplicazione avverrà tra due matrici, di cui una rappresenta l'immagine oggetto di analisi e l'altra il filtro che viene applicato.
Nel processo di convoluzione si individuano tre elementi fondamentali:

- *Immagine di input*: una matrice di pixel rappresentativa dell'immagine;
- *Feature detector* o *Kernel*: una matrice, in generale di dimensioni minori rispetto all'immagine, che agisce da filtro attraverso l'operazione di convoluzione;
- *Feature map*: una matrice risultante dalla convoluzione tra immagine di input e kernel.

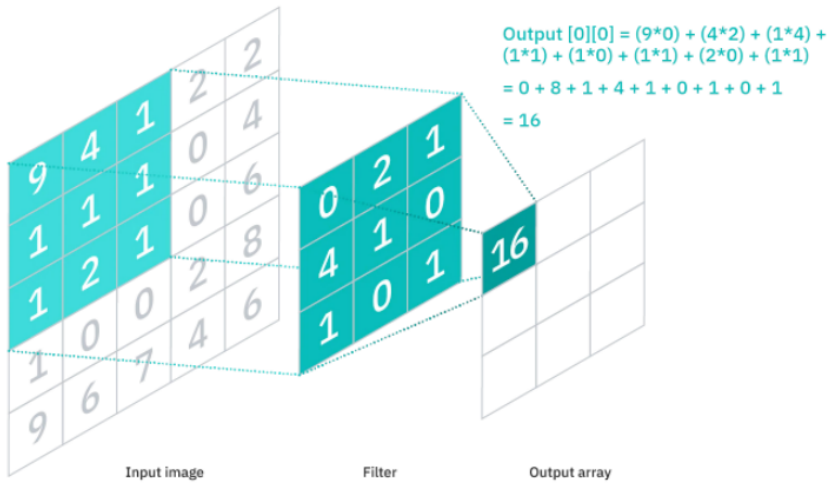

Figura 3.8. Rappresentazione dell'operazione di convoluzione

In sintesi, si ha un'immagine di input e un filtro, che è un insieme di pesi, il quale viene applicato sistematicamente ai dati di input per creare una mappa delle caratteristiche. Nel processo di convoluzione si perdono alcune informazioni sull'immagine, ossia quelle

non contenute nella feature detector prescelta; tuttavia, utilizzando più filtri di convoluzione si otterranno molteplici feature map, ciascuna delle quali memorizzerà caratteristiche distintive dell'immagine.
Come si nota in *Figura 3.8*, ogni valore di output nella mappa delle caratteristiche non deve connettersi a ciascun valore di pixel nell'immagine di input, ma solo al suo campo recettivo. Per questa ragione, i livelli convoluzionali sono comunemente indicati come livelli "parzialmente connessi". Si noti, inoltre, che i pesi nel kernel rimangono fissi mentre si sposta sull'immagine, operazione nota anche come condivisione dei parametri.
Nella definizione di un layer convoluzionale i parametri in gioco sono i seguenti:

- *Kernel size*, ovvero le dimensioni della matrice del kernel di convoluzione;
- *Padding*: rappresenta le dimensioni (in pixel) del bordo esterno applicato all'immagine di input. La tecnica che consiste nell'aggiungere un "bordo di zeri" all'immagine è definita *zero padding*. Considerando che l'applicazione di un filtro riduce le dimensioni dell'input, questa tecnica permette di preservare la dimensione dell'immagine in uscita dal layer per non perdere informazioni;
- *Stride*: rappresenta la velocità di traslazione del kernel sull'immagine, misurata in pixel orizzontali e verticali. Utilizzare un passo maggiore permette di ridurre la dimensione delle feature map nel volume di output e conseguentemente il numero di connessioni;
- *Numero di filtri*: influisce sulla profondità dell'output. Ogni filtro produce un output che si concatena con l'output prodotto dagli altri filtri. Questa concatenazione rappresenta l'input per il successivo layer.

Dopo ogni operazione di convoluzione, una CNN introduce la non linearità nel modello applicando una funzione di attivazione alle varie feature maps.
Come visto in precedenza, la funzione di attivazione storicamente più utilizzata è la sigmoide. Tuttavia, nell'addestramento delle reti profonde tramite retropropagazione dell'errore in combinazione con SGD il suo utilizzo diventa problematico: essa può causare il *problema della scomparsa del gradiente* (o *vanishing gradient problem*).
Per comprendere l'entità di tale problema facciamo un passo indietro. Ricordando che l'algoritmo di Error Back-propagation è basato sulla regola della catena, è possibile rappresentare il gradiente della funzione di costo come prodotto dei gradienti di tutte le funzioni di attivazione dei nodi rispetto ai loro pesi. Pertanto, i pesi aggiornati dei nodi nella rete dipendono dai gradienti delle funzioni di attivazione di ciascun nodo.
La derivata parziale della funzione sigmoidea raggiunge un valore massimo di 0.25. Quando ci sono più strati nella rete, man mano che ci si sposta indietro il prodotto dei gradienti tende a ridursi fino a quando a un certo punto si avvicina a un valore prossimo allo zero. Di conseguenza, man mano che la derivata svanisce i pesi della rete

rimangono invariati impedendo alla rete di apprendere.
Risulta evidente che il problema della scomparsa del gradiente è causato dalla derivata della funzione di attivazione utilizzata per creare la rete neurale. Pertanto, può essere affrontato usando differenti funzioni di attivazione, come ad esempio ReLU. Le unità lineari rettificate (ReLU) sono funzioni di attivazione che generano un'uscita lineare positiva quando vengono applicate a valori di ingresso positivi. Se l'input è negativo, la funzione restituirà zero. Il vantaggio nel suo utilizzo per l'attivazione in una rete neurale al posto di una funzione sigmoidea risiede nel fatto che il valore della derivata parziale della funzione di costo varrà 0 o 1, impedendo al gradiente di svanire.

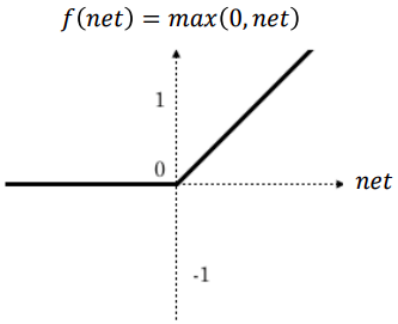

Figura 3.9. Funzione di attivazione ReLU

Strato di pooling o Pooling layer

Tipicamente ogni layer convoluzionale viene fatto seguire da uno di pooling, noto anche come downsampling layer. Il suo compito consiste nel ridurre l'immagine di input al fine di limitare il carico computazionale, l'utilizzo della memoria e il numero di parametri (limitando così il rischio di overfitting).
Un livello di pooling esegue un'aggregazione delle informazioni nel volume di input, generando feature map di dimensione inferiore. L'aggregazione opera generalmente nell'ambito di ciascuna feature map, in modo tale da far rimanere invariato il numero di mappe nel volume di input e di output. Similmente a quanto avviene nel livello convoluzionale, l'operazione di aggregazione fa scorrere un filtro sull'intero input, ma la differenza è che questo filtro non ha pesi. In altre parole, un filtro di pooling è simile a un filtro di convoluzione, con una propria dimensione e stride, ma non ha parametri da apprendere.
Esistono due tecniche di pooling:

- *Max pooling*: quando il filtro si sposta sull'input e seleziona il pixel con il valore massimo da inviare all'array di output. Questo approccio tende ad essere utilizzato più spesso rispetto all'average pooling;
- *Average pooling*: quando il filtro si sposta sull'input e calcola il valore medio all'interno del campo recettivo da inviare all'array di output.

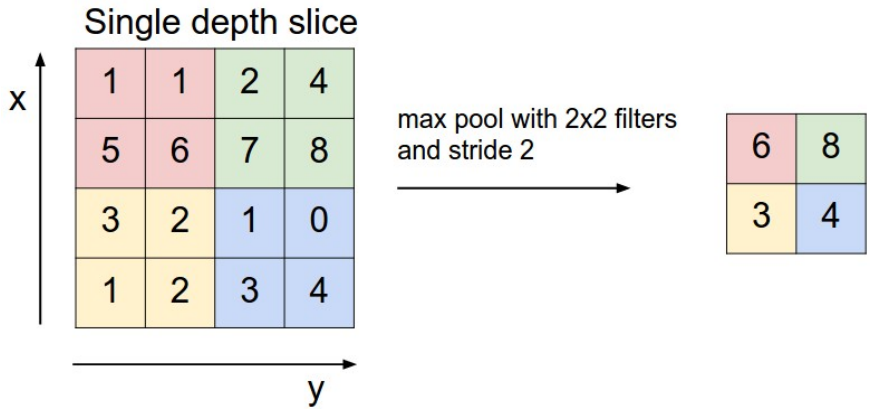

Figura 3.10. Esempio di un max-pooling con Filtri 2 x 2 e Stride = 2.

Strato completamente connesso o Fully-Connected Layer

Dopo n processi di Convoluzione - ReLU - Pooling, occorrerà trasformare le pooled feature maps in un vettore a una dimensione, il quale sarà l'input della rete fully connected. Il processo che permette questa trasformazione prende il nome di *Flattening* (appiattimento).
Successivamente, il livello fully-connected esegue il compito di classificazione in base alle caratteristiche estratte attraverso i livelli precedenti e ai loro diversi filtri. Esso emetterà un vettore di dimensioni k, dove k è il numero di classi che la rete sarà in grado di prevedere, contenente le probabilità per ciascuna classe di qualsiasi immagine classificata.
Al termine del processo si giungerà al livello di output, dove verranno applicate due tipologie di funzioni di attivazione per completare la classificazione dell'immagine producendo una probabilità da 0 a 1: la funzione sigmoidea, in caso di classificazione binaria, oppure la sua variante multi-dimensionale chiamata softmax, adatta in caso di classificazione relativa a molteplici categorie.

Capitolo 4

Caso Studio (parte 1): Faster R-CNN

Come anticipato nel *Capitolo 2*, una delle principali caratteristiche della tecnica AE è la capacità di localizzare il danno, consentendo una comprensione preliminare della possibile area danneggiata. Per definire con accuratezza la posizione della sorgente risulta fondamentale la determinazione affidabile ed esatta del tempo di inizio.
Dopo aver fatto una disamina delle Reti Neurali Artificiali, ponendo particolare attenzione alle Reti Neurali Convoluzionali, in questo capitolo verrà utilizzata una CNN con il fine di rilevare in maniera automatica il tempo di insorgenza (*Onset time*) dei segnali di Emissione Acustica.
Generalmente non è una buona idea addestrare una Deep Neural Network complessa, ovvero con molti pesi e connessioni, da zero principalmente per tre motivi:

- L'addestramento può richiedere molto tempo (giorni, o addirittura, settimane anche con l'utilizzo di GPU);
- L'addestramento può essere molto costoso dal punto di vista computazionale;
- L'addestramento su un nuovo problema richiede un set di dati etichettati di grandi dimensioni, spesso non disponibile o, comunque, molto costoso da ottenere.

Un'alternativa valida al training da zero risiede nella tecnica del Transfer Learning, o apprendimento per trasferimento.

4.1 Transfer Learning

Il termine *Transfer Learning* denota un avanzato metodo di Machine Learning che consente di riutilizzare gran parte dei parametri (pesi) di un modello pre-addestrato su un problema simile a quello da risolvere, mediante l'utilizzo di uno o più livelli dello stesso. Un modello pre-addestrato è una rete salvata che è stata precedentemente addestrata su un set di dati di grandi dimensioni, in genere su un'attività di classificazione delle immagini su larga scala. Esso può essere riutilizzato nella sua interezza o utilizzato come base per personalizzare un altro modello di rete neurale a svolgere una determinata attività. L'intuizione alla base del Transfer Learning è che se un modello viene addestrato su un set di dati sufficientemente ampio e generale, questo fungerà effettivamente da modello generico del mondo visivo. È quindi possibile sfruttare queste mappe delle funzionalità apprese senza dover iniziare da zero addestrando un modello complesso su un set di dati di grandi dimensioni [28].
Come indicato in *Figura 4.1*, della rete neurale pre-addestrata si fissano solo i pesi relativi agli strati inferiori. L'addestramento, dunque, si soffermerà sugli ultimi layer che sono solitamente quelli dedicati alla classificazione e/o alla regressione delle feature ottenute con i layer precedenti.

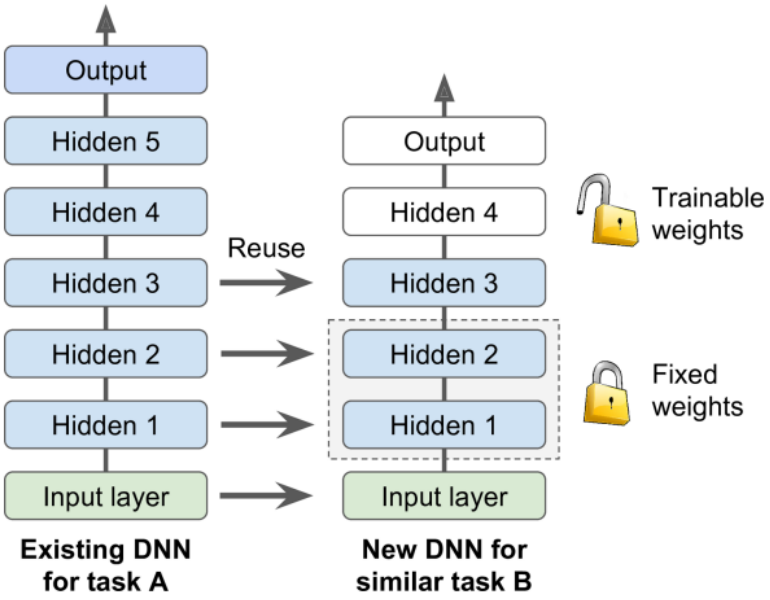

Figura 4.1. Transfer Learning: riutilizzo di strati pre-addestrati [7]

In particolare, lo strato di output del modello originale viene di solito sostituito perché molto probabilmente non è utile per il nuovo compito e potrebbe anche non avere il giusto numero di uscite. Allo stesso modo, gli strati nascosti superiori del modello originale hanno meno probabilità di essere utili, poiché diventano progressivamente più specifici per i dettagli delle classi contenute nel dataset originale.
In linea di principio, più dati di allenamento si hanno, maggiore sarà il numero di strati

che si possono "sbloccare" [7].
Più precisamente, esistono due modi principali per personalizzare un modello pre-addestrato:

1. ***Feature Extraction***. In questa ipotesi non è necessario riaddestrare l'intero modello, in quanto utilizza le rappresentazioni apprese da una rete precedente per estrarre funzionalità significative da nuovi campioni. Ciò che fa è aggiungere semplicemente in cima al modello pre-addestrato un nuovo classificatore, che verrà addestrato da zero, in modo da poter riutilizzare le mappe delle caratteristiche apprese in precedenza per il set di dati;

2. ***Fine-Tuning***. Questa seconda ipotesi permette di limitare l'addestramento e l'elaborazione a un numero sensibilmente inferiore di parametri. In aggiunta al nuovo classificatore, si sbloccano anche alcuni dei livelli superiori del modello pre-addestrato. In tal modo si consente di "perfezionare" le rappresentazioni delle caratteristiche di livello superiore presenti nel modello base per renderle più rilevanti e conformi all'esecuzione del nuovo compito. Gli strati riutilizzati sono etichettati come "read-only".

Risulta, dunque, chiaro come il Transfer Learning permetta di velocizzare i tempi di addestramento, diminuire la potenza di elaborazione richiesta e incrementare l'accuratezza della rete a fronte di un set di dati di allenamento significativamente inferiore.
È possibile trovare diversi modelli pre-addestrati su dataset standardizzati. Tra i dataset più famosi troviamo ImageNet [29] (che contiene 1,4 milioni di immagini con 1000 categorie) e COCO (*Common Objects in Context*) [30], ma ne esistono molti altri a seconda dello scopo per cui quella rete neurale è stata progettata.

4.2 Stato dell'Arte

Per il rilevamento automatico dell'Onset Time si è messa a punto (*fine tune*) una Faster R-CNN pre-addestrata con COCO [21], utilizzando Pytorch.
Pytorch [31] è una tra le più popolari librerie del linguaggio Python (assieme a Keras e TensorFlow) utilizzate nel campo del Deep Learning [7]. Precisamente, il modulo contiene diversi metodi e funzioni scientifiche per le applicazioni dedicate al Machine learning e al Deep Learning. Tra i suoi punti di forza vi è la possibilità di elaborare i tensori (array multidimensionali) utilizzando l'accelerazione della GPU, aspetto che la renda più veloce rispetto alla libreria Numpy.
Faster R-CNN è una rete convoluzionale profonda utilizzata per il rilevamento di oggetti, che può prevedere in modo accurato e rapido la posizione di oggetti diversi.
Per comprendere appieno la Faster R-CNN, risulta necessaria una rapida panoramica delle reti da cui si è evoluta, ovvero R-CNN e Fast R-CNN.

4.2.1 R-CNN

Nel 2014, un gruppo di ricercatori dell'UC Berkely ha sviluppato una rete convoluzionale profonda chiamata R-CNN (Region-based Convolutional Neural Network) [8]. Tale sistema di Object Detection è costituito da 3 fasi:

1. ***Region proposals***. Il primo modulo prende un'immagine in input ed estrae circa 2000 proposte regionali, indipendenti dalla categoria, utilizzando l'algoritmo di ricerca selettiva [22]. Queste proposte definiscono l'insieme di candidati disponibili per il rilevatore;

2. ***Feature extraction***. Il secondo modulo estrae un vettore di caratteristiche di lunghezza 4.096 da ciascuna proposta di regione. Le caratteristiche sono calcolate propagando in avanti un'immagine attraverso cinque livelli convoluzionali e due livelli completamente connessi. Per fare ciò, è necessario prima convertire i dati dell'immagine in quella regione in una forma compatibile con la CNN (la sua architettura richiede input di una dimensione fissa di 227x227 pixel);

3. ***Classify Regions***. Il terzo modulo utilizza un algoritmo SVM pre-addestrato per classificare la proposta della regione in una delle classi di oggetti.

In altre parole, prima vengono generate le possibili regioni, quindi si estraggono le caratteristiche e infine si classificano tali regioni sulla base delle caratteristiche estratte. In sostanza, tale sistema trasforma la rilevazione di oggetti in un problema di classificazione.
Come sottolineato nell'articolo [8], due proprietà rendono efficiente il rilevamento.
In primo luogo, tutti i parametri CNN sono condivisi tra tutte le categorie. In secondo luogo, i vettori delle caratteristiche calcolati dalla CNN sono a bassa dimensione (low dimensional) rispetto ad altri approcci comuni. Il risultato di tale condivisione è che il

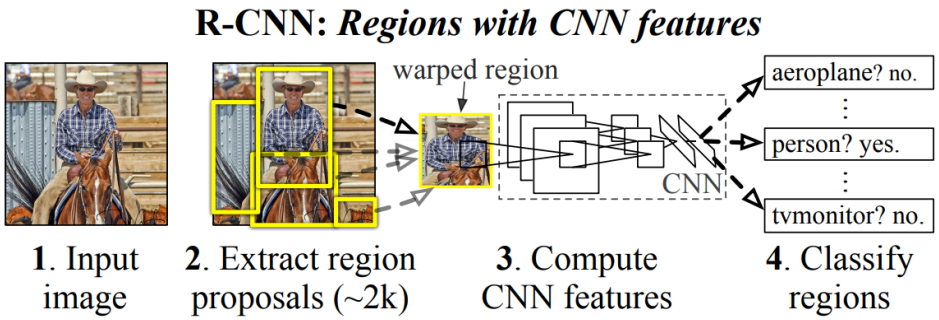

Figura 4.2. Panoramica del sistema di Object Detection R-CNN [8]

tempo trascorso a elaborare le proposte e le caratteristiche della regione (13 s/image su una GPU o 53 s/image su una CPU) è ammortizzato su tutte le classi.
Sebbene tale metodo raggiunga un'eccellente accuratezza nel rilevamento degli oggetti, presenta notevoli svantaggi [9]:

- L'addestramento è un processo in più fasi;
- L'addestramento è costoso nello spazio e nel tempo. Il metodo memorizza le funzionalità estratte dalla CNN pre-addestrata sul disco per addestrare in seguito le SVM. Ciò richiede centinaia di gigabyte di spazio di archiviazione;
- Il processo di rilevamento degli oggetti è molto lento

4.2.2 Fast R-CNN

Nel 2015 Ross Girshick propone il modello Fast R-CNN [9] come estensione del modello R-CNN. Il nuovo algoritmo di allenamento risolve alcuni svantaggi di R-CNN, migliorandone velocità e precisione. Il termine Fast vuole sottolineare come si tratti di un metodo relativamente veloce da addestrare e testare.
Tra i vantaggi di tale metodo troviamo:

- Maggiore qualità di rilevamento rispetto a R-CNN;
- È costituito da uno stadio singolo, rispetto ai 3 stadi in R-CNN;
- L'addestramento può aggiornare tutti i livelli della rete;
- Non memorizza nella cache le funzionalità estratte; di conseguenza, non ha bisogno di così tanto spazio di archiviazione su disco come invece accade per la R-CNN.

L'architettura generale di Fast R-CNN è mostrata di seguito:
La rete prende in input un'immagine e un insieme di regioni di interesse (RoIs). In primis, la rete elabora l'intera immagine con diversi livelli convoluzionali e di max pooling

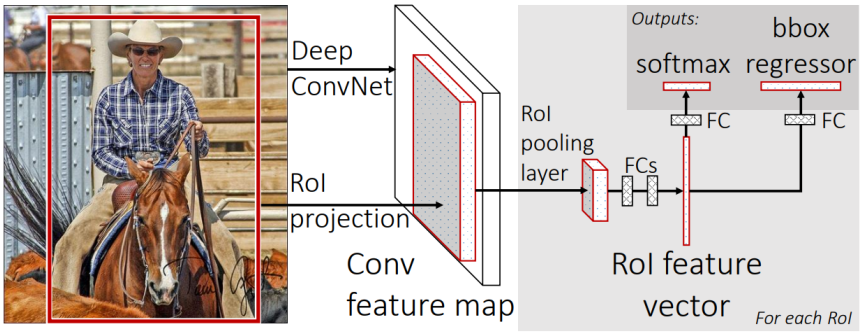

Figura 4.3. Architettura Fast R-CNN [9]

per produrre una feature map. Questa viene inviata a un nuovo livello chiamato *RoI Pooling layer*, con il fine di estrarre vettori di caratteristiche di uguale lunghezza da tutte le proposte (cioè le RoI) nella stessa immagine.
Con RoI (Region of Interest) si indica una finestra rettangolare all'interno di una feature map, definita attraverso quattro parametri (r,c,h,w): vertice superiore sinistro (r, c), altezza (h) e larghezza (w). Il RoI Pooling layer utilizza il max pooling per convertire le features all'interno di qualsiasi regione di interesse valida in una feature map più piccola con dimensione fissa HxW, dove H e W sono iperparametri indipendenti da qualsiasi RoI.
Il risultato di questo processo viene conseguentemente convertito in un vettore di caratteristiche (RoI feature vector) attraverso strati completamente connessi (FCs). Infine, il vettore di caratteristiche estratto viene passato ad alcuni livelli FC che si diramano in due livelli di output: uno strato softmax che produce stime di probabilità sulle classi di oggetti e un altro livello che produce i riquadri di delimitazione degli oggetti rilevati.
Nonostante i miglioramenti apportati da tale metodo, persiste una criticità: l'utilizzo dell'algoritmo di ricerca selettiva per generare proposte regionali richiede tempo. Inoltre, tale algoritmo non può essere personalizzato su un'attività di rilevamento di oggetti specifica. Pertanto, potrebbe non essere sufficientemente accurato per rilevare tutti gli oggetti nel set di dati.

4.2.3 Faster R-CNN

Nel 2015, poco dopo la pubblicazione di Fast R-CNN, un gruppo di ricercatori di Microsoft ne propone un'estensione che prende il nome di Faster R-CNN [10].
L'intuizione è stata quella di rimpiazzare il lento algoritmo di ricerca selettiva con una rete neurale veloce. Nello specifico essa ha introdotto la Region Proposal Network (RPN o Rete di Proposta Regionale).
La Faster R-CNN è composta da due moduli:

1. ***Region Proposal Network*** (RPN): per la generazione di proposte regionali;
2. ***Fast R-CNN*** : per rilevare oggetti nelle regioni proposte.

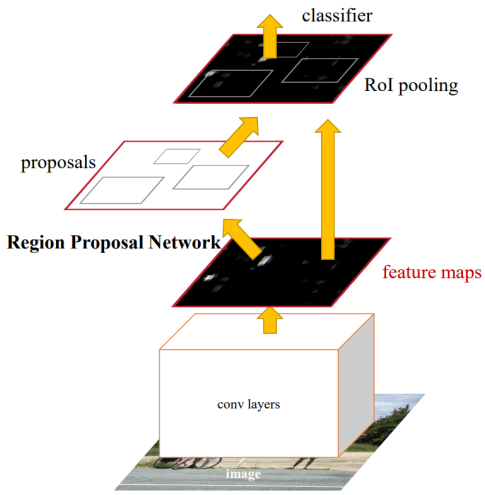

Figura 4.4. Architettura Faster R-CNN [10]

L'intero sistema rappresenta una rete unificata per il rilevamento di oggetti, in cui l'RPN implementa il concetto di 'attenzione' nelle reti neurali per indicare al modulo di rilevamento Fast R-CNN dove cercare gli oggetti nell'immagine. Ciò che permette di unificare i due moduli in un'unica rete è la condivisione degli stessi livelli convoluzionali. Tale condivisione consente, inoltre, proposte di regioni quasi a costo zero.

Region Proposal Network

Ciò che fa una Rete di Proposta Regionale è prendere in input un'immagine e restituire un set di proposte di oggetti in forma rettangolare. Ad ogni proposta, inoltre, viene associato un 'punteggio di oggettività' (*objectness score*), il quale permette di valutare l'appartenenza ad una classe di oggetti piuttosto che allo sfondo.
Questo processo è modellato attraverso una rete convoluzionale che, come sottolineato in precedenza, condivide i layer convoluzionali con Fast R-CNN.
Per generare le cosiddette 'proposte' per la regione in cui si trova l'oggetto, una piccola rete viene fatta scorrere sulla feature map di output restituita dall'ultimo strato di convoluzione condiviso. Per ogni posizione della finestra scorrevole vengono generate contemporaneamente diverse proposte di regioni candidate, le quali verranno in seguito filtrate in base al loro punteggio di oggettività. Il numero massimo possibile di proposte per ogni posizione è indicato con la lettera k. Tali proposte sono basate su vincoli spaziali di dimensioni fisse chiamati riquadri di ancoraggio (*anchor boxes*). Un ancoraggio è centrato nella finestra scorrevole in questione, ed è associato ad una scala e ad una proporzione. Così facendo è possibile utilizzare una singola immagine su una singola scala, in quanto gli ancoraggi esistono su scale diverse. In altre parole, la presenza di ancoraggi multiscala nell'algoritmo si traduce in '*Piramide di ancore*' invece di '*Piramidi di filtri*' (cioè filtri multipli con dimensioni diverse) o '*Piramidi di immagini*' (cioè istanze multiple dell'immagine ma a scale diverse), il che lo rende meno

dispendioso in termini di tempo e più efficiente in termini di costi rispetto agli algoritmi proposti in precedenza. Ciascuna regione viene quindi mappata su ciascun riquadro di ancoraggio di riferimento, rilevando così oggetti su scale e proporzioni diverse.
In seguito, per ogni proposta di regione viene estratto un vettore di caratteristiche, che verrà inviato a 2 strati fratelli completamente connessi:

- Il primo livello FC è denominato *cls* e rappresenta un classificatore binario che genera il punteggio di oggettività per ciascuna proposta di regione. In altre parole, cls valuta per ogni proposta la probabilità che vi sia o meno un oggetto e lo fa generando un vettore di 2 elementi. L'output prodotto da tale layer sarà, dunque, pari a 2k;
- Il secondo livello FC è denominato *reg* e restituisce un vettore 4D che definisce il riquadro di delimitazione della regione. L'output prodotto da tale layer sarà, dunque, pari a 4k;

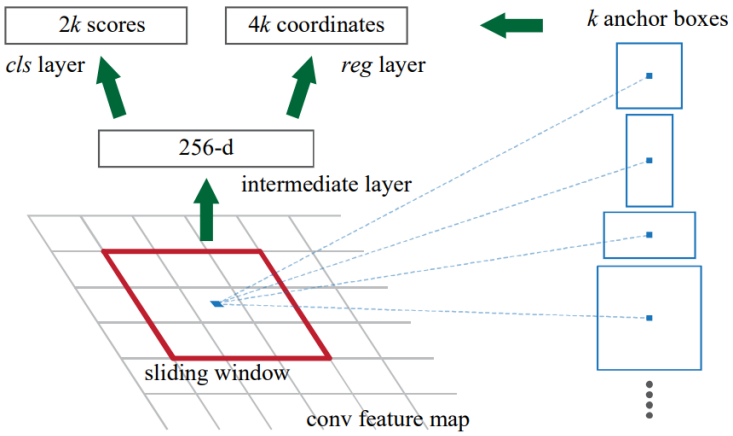

Figura 4.5. Region Proposal Network (RPN) [10]

È importante notare che, nonostante la RPN elabori le coordinate dei riquadri di delimitazione, essa non classifica comunque i possibili oggetti: a tale rete importa solo che, in effetti, sembrino oggetti e non sfondo.
Un'importante proprietà di questo approccio è che risulta essere invariante alle traslazionali (*translation invariant*), sia in termini di ancoraggi che di funzioni che calcolano proposte relative ad essi: se un oggetto viene traslato, la proposta dovrebbe traslare e la stessa funzione dovrebbe essere in grado di prevedere la proposta in entrambe le posizioni.
Per addestrare l'RPN, ad ogni ancoraggio viene assegnato un punteggio di oggettività positivo o negativo basato sulla metrica di valutazione '*Intersection-over-Union*'(IoU). Per valutare l'IoU si ha bisogno dei riquadri di delimitazione etichettati a mano (*ground-truth box*), che specificano dove si trova l'oggetto nell'immagine, e i riquadri di delimitazione previsti dal modello. L'IoU si valuta come il rapporto tra l'area di intersezione

tra il riquadro di delimitazione previsto e il ground-truth box e l'area di unione, o più semplicemente, l'area racchiusa dai due riquadri. Il suo valore varia da 0 (quando non c'è intersezione) a 1 (quando le due caselle coincidono) man mano che le due caselle si avvicinano l'una all'altra. Si assegnano etichette positive a due tipologie di ancoraggi:

- agli ancoraggi con l'IoU più alto con una ground-truth box;
- a un ancoraggio che ha un IoU superiore a 0.7 con qualsiasi ground-truth box;

Si assegnano, invece, etichette negative agli ancoraggi tali per cui l'IoU per tutte le ground-truth box è inferiore a 0.3. Un punteggio di oggettività negativo significa che l'ancoraggio è classificato come sfondo. Infine, ancoraggi che non sono né positivi né negativi non contribuiscono all'addestramento.
L'RPN può essere addestrata end-to-end attraverso retropropagazione e SGD.
Poiché gli ancoraggi di solito si sovrappongono, alcune proposte finiscono per sovrapporsi sullo stesso oggetto. Per ridurre la ridondanza viene applicata la *Non-Maximum Suppression* (NMS) sulle regioni proposte in base ai loro punteggi cls. NMS prende l'elenco delle proposte ordinate per punteggio e itera sull'elenco ordinato, scartando quelle proposte che hanno un IoU maggiore di una soglia predefinita con una proposta che ha un punteggio più alto. Fissando la soglia di IoU per l'NMS a 0.7, rimangono circa 2000 regioni proposte per immagine.
Dopo il passaggio RPN, si hanno numerose proposte di oggetti a cui non è stata assegnata alcuna classe. Avendo ora le possibili regioni, le introduciamo direttamente nella Fast R-CNN, la quale restituirà i punteggi di classe degli oggetti rilevati oltre ai relativi riquadri di delimitazione.

4.3 Analisi del modello utilizzato

Il punto di partenza per raggiungere lo scopo sopra citato è stato il codice contenuto all'interno di un notebook di Google Colab [32].
Tale codice è composto dai seguenti step:

1. ***Installazioni e importazioni***. In prima istanza vengono installate le librerie utili e si clona il TorchVision repo, il quale fornisce alcuni file di aiuto per la formazione, la valutazione e le trasformazioni;

2. ***Dataset***. Per la formazione di un modello accurato, è essenziale avere un buon set di dati: il secondo step consiste, dunque, nella sua importazione nel codice. Vengono create due cartelle contenenti rispettivamente il training set e il test set. Utilizzando un apprendimento supervisionato, ad ogni immagine contenuta nel dataset viene associato un file di testo (.txt) che contiene le informazioni utili per disegnare un riquadro attorno all'oggetto da identificare, secondo le seguenti disposizioni: *label, coordinate (x,y) baricentro oggetto, larghezza e altezza.*

3. ***Augmentation***. In questo step viene applicata la tecnica di augmentation, la quale permette di aumentare artificialmente la quantità di dati aggiungendo copie leggermente modificate dei dati contenuti nel dataset. Le tecniche di aumento dei dati per il rilevamento di oggetti variano rispetto a quelle utilizzate per le attività di classificazione, poichè in questi casi bisogna garantire che la bounding box si allinei ancora correttamente con l'oggetto dopo la trasformazione. Nel codice utilizzato viene impiegato il Vertical flip, il quale capovolge verticalmente l'immagine con una data probabilità (il valore predefinito è 0,5);

4. ***Preparazione del dataset***. In questa fase avviene lo splittaggio del training set, dal quale verrà preso un 20% di dati al fine di utilizzarli come validation set. Inoltre, viene caricato il set di dati in memoria utilizzando il dataloader, il quale combina un set di dati e un campionatore e fornisce un'iterazione sul set di dati specificato;

5. ***Modello pre-addestrato***. A questo punto si definisce una funzione per caricare il modello preaddestrato su COCO, la quale verrà richiamata in seguito. A tale modello si sostituisce il classificatore, in base ai numeri di classi del nuovo set di dati;

6. ***Training***. Questa è la fase in cui avviene l'addestramento. Si definisce un ottimizzatore (in questo caso SGD) e il numero di epoche;

7. ***Filtraggio degli output***. Poichè il modello prevede numerose bounding box sovrapposte, si utilizza la *Non-Maximum Suppression* (NMS);

8. ***Test del modello***. Infine, si prende un'immagine dal test set per fare una previsione, così da poter valutare il funzionamento del modello.

4.4 Applicazione della rete a un dataset di segnali sismici

Nel campo delle indagini sui meccanismi di frattura, esiste da più di trent'anni un'intensa cooperazione tra sismologi terrestri e ingegneri civili. Numerose procedure di analisi sismica e altri fenomeni sismologici sono stati trasferiti alle scienze dei materiali, poiché i parallelismi con l'analisi delle emissioni acustiche sono evidenti [23].
Un terremoto è un esempio di AE su larga scala (Katsuyama, 1994). Il meccanismo della generazione AE è lo stesso, sia esso un microcrack o un terremoto [24].
Dato l'evidente parallelismo tra i segnali di Emissione Acustica e quelli sismici, si è deciso di valutare le prestazioni del modello individuato per l'obiettivo prefissato utilizzando questi ultimi. Ricordando che l'addestramento di reti neurali artificiali richiede la presenza di grandi quantità di dati, la composizione del dataset è stata resa possibile grazie a ITACA (*ITalian ACcelerometric Archive*) [33], l'archivio italiano delle forme d'onda accelerometriche, acquisite prevalentemente dalla Rete Accelerometrica Nazionale (RAN), gestita dal Dipartimento della Protezione Civile (DPC), e dalla Rete Sismica Nazionale (RSN), gestita dall'Istituto Nazionale di Geofisica e Vulcanologia (INGV). Grazie ad esso, si ha accesso alla più completa collezione di registrazioni accelerometriche relative ad eventi sismici medio-forti occorsi in Italia. ITACA 3.1 contiene più di 50,000 forme d'onda relative a 2,052 eventi sismici di magnitudo superiore a 3.0, che si sono verificati nel periodo 1972-2019 e registrate da 1,498 stazioni.
Da questo archivio, si sono esportate le forme d'onda in termini di accelerazione (espressa in cm/s^2), le quali vengono fornite come file ASCII (formato DYNA 1.2). Per lavorare su di esse si è fatto uso del linguaggio di programmazione Python.

4.4.1 Primo tentativo

Una volta importate le forme d'onda accelerometriche, si è provveduto in prima battuta al loro plottaggio e conseguente salvataggio in formato JPG secondo le seguenti impostazioni:

- limitare l'asse delle ascisse, in modo tale che lo zero dell'asse dei tempi tocchi il bordo sinistro dell'immagine e il tempo relativo alla durata del segnale il bordo destro dell'immagine;
- limitare l'asse delle ordinate, in modo tale che vada da $-|ampiezza_{max}|$ a $|ampiezza_{max}|$;

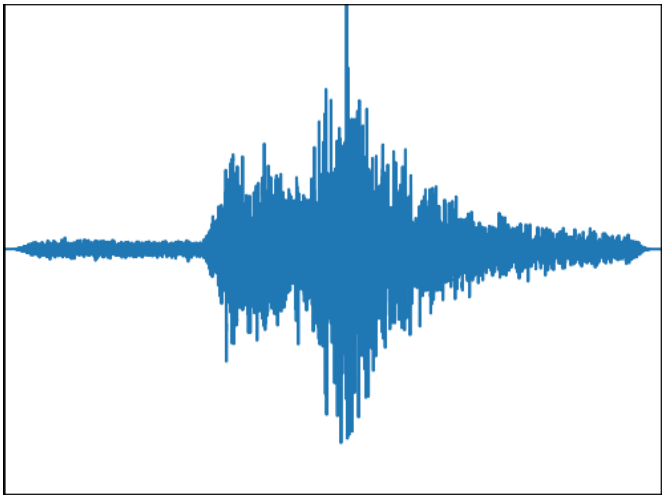

Figura 4.6. Esempio di forma d'onda accelerometrica contenuta nel dataset

Come sottolineato in precedenza, ad ogni immagine contenuta nel dataset viene associato un file di testo contenente le informazioni utili per disegnare un riquadro attorno all'oggetto da identificare. Nel caso in esame, l'idea è stata quella di vincolare il riquadro a passare in corrispondenza dell'Onset Time (punto che vogliamo determinare) e dell'ampiezza massima in valore assoluto. Quest'ultimo valore si è scelto così da poter permettere, in seguito ad un post processamento dell'output, informazioni aggiuntive circa la modalità di fessurazione (vedi 2.5).
Per etichettare i dati, dunque, è stato necessario individuare preventivamente l'Onset Time in maniera manuale.

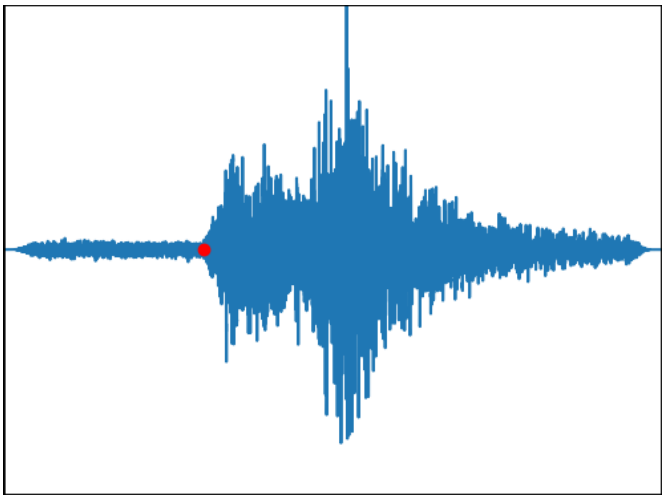

Figura 4.7. Individuazione manuale dell'Onset Time

In definitiva, il bounding box avrà un'altezza pari al doppio della $|y_{max}|$ (ampiezza massima del segnale) e una larghezza pari alla differenza tra il tempo in corrispondenza

del valore massimo e il tempo di inizio del segnale.

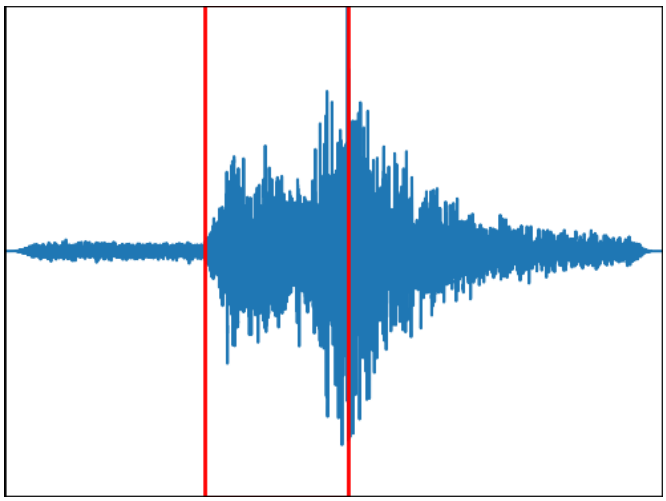

Figura 4.8. Rappresentazione del bounding box

A questo punto, avendo tutte le informazioni necessarie, si procede con la creazione del file TXT. Ricapitolando, le indicazioni contenute al suo interno sono:

label *coordinate (x,y) centro box* *larghezza box* *altezza box*

Nel caso in esame:

- la label (ovvero l'etichetta) è pari a 0 per tutte le immagini contenute nel dataset, in quanto ciò che si andrà ad identificare sarà sempre una parte di un segnale;
- la coordinata x del centro del riquadro sarà pari a:

$$X_{centro} = Onset\ Time + \frac{larghezza\ box}{2} \tag{4.1}$$

All'interno del file di testo, si andrà a riportare tale coordinata espresse in pixel e normalizzata rispetto alla dimensione dell'immagine. Essendo quest'ultima pari a 496x369, le trasformazioni utilizzate sono:

$$X_{centro,secondi} : durata\ del\ segnale = X_{centro,pixel} : 496 \tag{4.2}$$

$$X_{centro,pixel} = \frac{X_{centro,secondi}}{durata\ del\ segnale} \cdot 496 \tag{4.3}$$

La coordinata espressa in pixel e normalizzata sarà dunque pari a :

$$\bar{X}_{centro,pixel} = \frac{X_{centro,pixel}}{496} = \frac{X_{centro,secondi}}{durata\ del\ segnale} \tag{4.4}$$

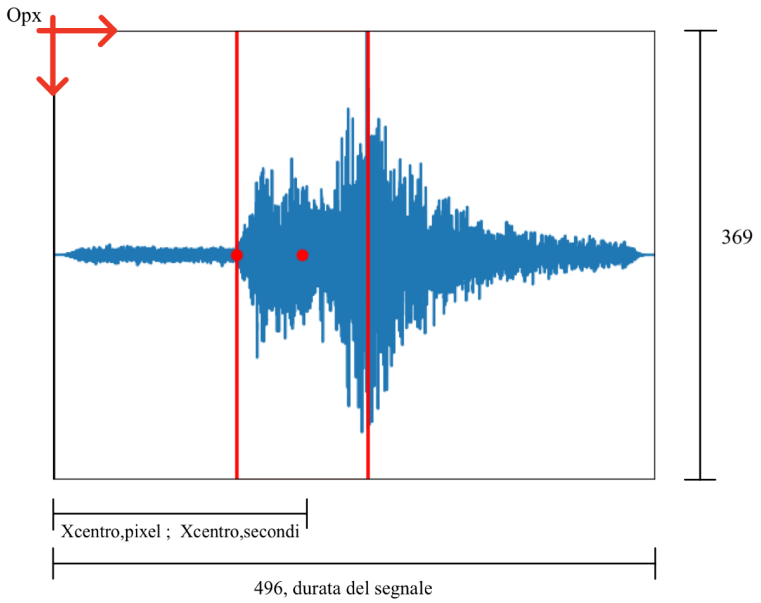

Figura 4.9. Passaggio in coordinate pixel e normalizzate

- la coordinata y del centro del riquadro espressa in pixel e normalizzata sarà pari a 0.5, in quanto si trova esattamente a metà dell'immagine (in corrispondenza del punto di inizio del segnale l'ampiezza è nulla).
- la larghezza del box espressa in pixel e normalizzata si ottiene seguendo le medesime trasformazioni utilizzate per la coordinata x;
- l'altezza del box espressa in pixel e normalizzata sarà pari a 1, essendo coincidente con l'altezza dell'immagine per le scelte effettuate.

In sintesi, il file di testo sarà del tipo:

$$0 \qquad \bar{X}_{centro,pixel} \qquad 0{,}5 \qquad \bar{W} \qquad 1$$

Partendo da un dataset contenente 410 forme accelerometriche, si sono create due cartelle aventi rispettivamente 328 immagini ($\sim$ 80 % del dataset) per il training set e 82 immagini ($\sim$ 20 % del dataset) per il test set. Ad ognuna di esse è stato associato il rispettivo file di testo per l'etichettamento del dato.
Una volta che il dataset è stato formato, esso viene importato nel codice: a questo punto è possibile procedere con l'addestramento della rete. L'output restituito dalla rete è del tipo:

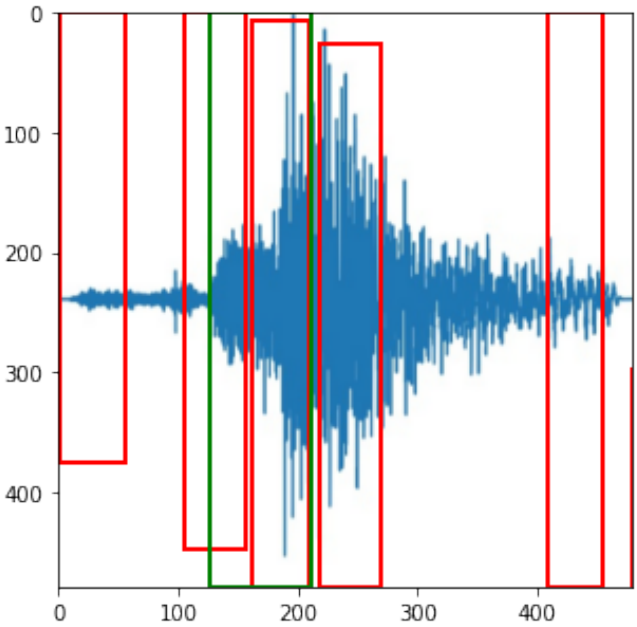

Figura 4.10. Esempio di output restituito dalla rete a seguito del primo tentativo: in verde la ground-truth box e in rosso le predizione della rete

Come emerge in figura, il primo tentativo effettuato non ha portato a risultati soddisfacenti.

4.4.2 Secondo tentativo

Dati gli scarsi risultati ottenuti a seguito del primo tentativo, l'idea è stata quella di aiutare la rete nella predizione del bounding box imponendo un'ulteriore limitazione nel plottaggio delle forme d'onda accelerometriche. La limitazione aggiuntiva consiste nel forzare a zero l'ampiezza del segnale subito dopo aver raggiunto $|ampiezza_{max}|$.

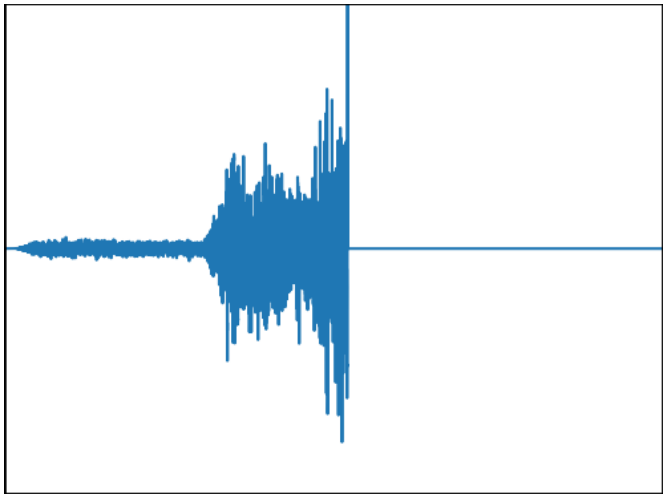

Figura 4.11. Esempio di segnale contenuto nel dataset del secondo tentativo

Non avendo cambiato nulla in merito alla composizione del bounding box, i file di testo creati precedentemente per l'etichettamento dei dati risultano tutt'ora riutilizzabili. Una volta riformato il dataset con le nuove immagini, è possibile procedere con l'addestramento della rete. A seguito di vari tentativi, si è arrivati al risultato più accurato utilizzando 20 epoche. L'output restituito dalla rete è del tipo:

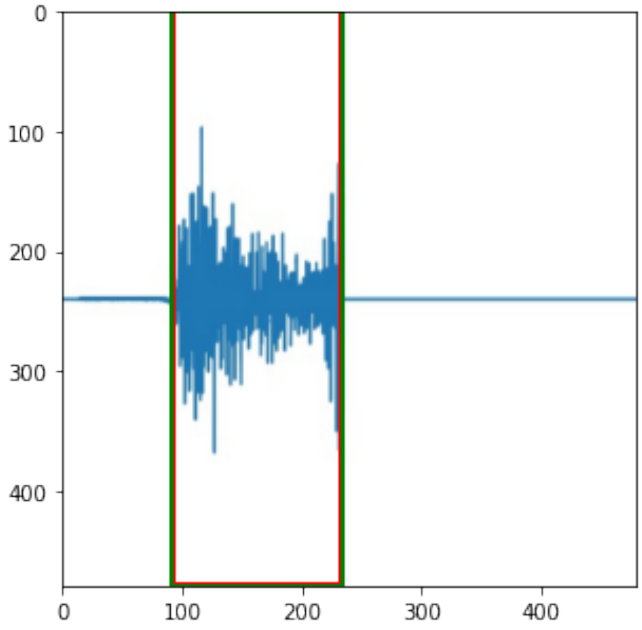

Figura 4.12. Esempio di output restituito dalla rete a seguito del secondo tentativo: in verde la ground-truth box e in rosso la predizione della rete

```
{'boxes': tensor([[ 92.9167,   0.0000, 232.8496, 477.7464]], device='cuda:0'),
 'labels': tensor([1], device='cuda:0'),
 'scores': tensor([0.9993], device='cuda:0')}
```

Figura 4.13. Esempio di output restituito dalla rete

dove:

1. la prima riga mostra le informazioni utili alla ricostruzione della bounding box. Per risalire alle seguenti informazioni:

   coordinate (x,y) del vertice in alto a sinistra *larghezza* *altezza*

   ciò che bisogna fare è:

   boxes[0] *boxes[1]* *boxes[2]-boxes[0]* *boxes[3]-boxes[1]*

2. la seconda riga rappresenta l'etichetta. Nel caso in esame si hanno due classi: lo sfondo (associato al valore 0) e il segnale (associato al valore 1);

3. la terza riga riporta il punteggio sulla base del quale viene applicata la Non-Maximum Suppression (NMS) per ridurre la ridondanza delle regioni proposte dalla rete.

Per come è stato impostato il problema, l'infomazione sull'Onset Time è racchiusa nel primo valore contenuto nella prima riga dell'output (ovvero 'boxes[0]').
Per confrontare tale valore con il target, i passaggi da seguire sono:

1. normalizzare il valore restituito dalla rete in base alle dimensioni dell'immagine. Nel codice utilizzato, durante il caricamento dei dati, le immagini sono state ridimensionate: si passa da una dimensione di partenza di 496x369 a una dimensione di 480x480. Indicando con $\hat{X}$ l'output della rete, l'equazione che ci permette di ottenere la coordinata espressa in pixel e normalizzata è:

$$\bar{X}_{Onset\ Time} = \frac{\hat{X}}{480} \tag{4.5}$$

2. passare da coordinata espressa in pixel e normalizzata a coordinata espressa in secondi invertendo l'equazione 4.4.

Di seguito si riporta la tabella di confronto tra gli output della rete, ottenuti impostando un numero di epoche pari a 20, e il target, ottenuto individuando manualmente l'Onset Time. Inoltre, per ogni segnale contenuto nel test set si valuta l'errore percentuale:

$$Errore\ \% = \frac{\bar{\hat{X}} - X}{X} \cdot 100 \tag{4.6}$$

Utilizzando come parametri di valutazione delle prestazioni della rete l'errore quadratico medio (MSE, vedi 3.1) e l'errore percentuale medio, si ottiene:

MSE	ERRORE % MEDIO
0.3026 s^2	3.54 %

Tabella 4.1. Valori delle metriche di valutazione ottenuti con la Faster R-CNN

Tabella 4.2: Confronto tra target e output della rete Faster R-CNN

N°	Output (pixel)	Output normalizzato	Durata segnale	Output (secondi)	Target	Δ^2	Errore %
0	43.8386	0.0913	76.85	7.0187	6.6766	0.1171	5.3
1	142.6735	0.2972	112.8	33.5283	34.3123	0.6146	2.3
2	41.9247	0.0873	76.85	6.7123	6.6663	0.0021	0.7
3	190.1885	0.3962	43.88	17.3864	17.6947	0.0950	1.7
4	180.456	0.3760	69.00	25.9406	25.7180	0.0495	0.9
5	180.7268	0.3765	45.98	17.3121	17.3917	0.0063	0.5
6	113.7511	0.2370	33.88	8.0289	8.0809	0.0027	0.6
7	175.3806	0.3654	59.98	21.9153	21.9424	0.0007	0.1
8	34.9621	0.0728	80.18	5.8401	6.2462	0.1649	6.5
9	161.2068	0.3358	64.00	21.4942	21.3388	0.0242	0.7
10	130.9068	0.2727	77.83	21.2260	21.2586	0.0011	0.2
11	168.2678	0.3506	45.98	16.1187	16.1888	0.0049	0.4
12	183.361	0.3820	55.98	21.3845	21.2394	0.0211	0.7
13	66.6801	0.1389	52.04	7.2292	7.2648	0.0013	0.5
14	19.27	0.0401	75.00	3.0109	3.3342	0.1045	9.7
15	113.4456	0.2363	69.99	16.5418	16.1084	0.1878	2.7
16	103.5900	0.2158	98.31	21.2165	21.4208	0.0417	1.0
17	98.5024	0.2052	95.4	19.5774	19.7650	0.0352	0.9
18	79.5395	0.1657	66.72	11.0560	11.4942	0.1920	3.8
19	94.7572	0.1974	85.03	16.7858	16.5483	0.0564	1.4

(Continua alla pagina successiva)

(Continua dalla pagina precedente)

20	29.5323	0.0615	108.3	6.6632	6.3929	0.0731	4.2
21	47.896	0.0998	64.96	6.4819	6.4869	0.0000	0.1
22	8.1556	0.0170	46.00	0.7816	0.7498	0.0010	4.2
23	1.29E+02	0.2693	69.99	18.8506	18.8218	0.0008	0.2
24	58.6054	0.1221	65.00	7.9361	7.2168	0.5174	10.0
25	92.9167	0.1936	90	17.4219	17.2736	0.0220	0.9
26	28.2784	0.0589	111.6	6.5747	7.0559	0.2315	6.8
27	78.7588	0.1641	155.9	25.5802	26.7501	1.3687	4.4
28	52.056	0.1085	105	11.3873	11.7133	0.1063	2.8
29	145.1506	0.3024	111.1	33.5963	34.2534	0.4318	1.9
30	149.9815	0.3125	111.7	34.9019	34.5188	0.1468	1.1
31	91.7154	0.1911	64.97	12.4141	12.7141	0.0900	2.4
32	152.132	0.3169	44.99	14.2592	13.9195	0.1154	2.4
33	131.0464	0.2730	31.96	8.7255	8.7702	0.0020	0.5
34	126.7139	0.2640	49.99	13.1967	13.3461	0.0223	1.1
35	173.5577	0.3616	64.96	23.4881	23.5893	0.0102	0.4
36	123.27	0.2568	62.36	16.0148	15.9862	0.0008	0.2
37	136.8224	0.2850	53.97	15.3840	15.3573	0.0007	0.2
38	229.5197	0.4782	80.74	38.6071	39.1341	0.2777	1.3
39	124.5198	0.2594	28.00	7.2637	7.4119	0.0220	2.0
40	131.6302	0.2742	31.96	8.7644	8.6965	0.0046	0.8
41	173.1122	0.3607	64.96	23.4279	23.5801	0.0232	0.6

(Continua alla pagina successiva)

(Continua dalla pagina precedente)

42	192.4444	0.4009	99.97	40.0806	40.5800	0.2495	1.2
43	104.7816	0.2183	54.97	11.9997	12.1891	0.0359	1.6
44	102.0621	0.2126	31.95	6.7935	6.7173	0.0058	1.1
45	170.5545	0.3553	35.03	12.4469	12.4685	0.0005	0.2
46	91.0625	0.1897	64.97	12.3257	12.4875	0.0262	1.3
47	104.1755	0.2170	50.04	10.8603	10.8104	0.0025	0.5
48	124.834	0.2601	28	7.2820	7.3759	0.0088	1.3
49	69.2958	0.1444	24.04	3.4706	3.4756	0.0000	0.1
50	161.6746	0.3368	79.97	26.9357	27.4446	0.2591	1.9
51	108.0701	0.2251	32.99	7.4276	7.5435	0.0134	1.5
52	151.1797	0.3150	73.45	23.1336	23.4023	0.0722	1.1
53	94.5692	0.1970	39.98	7.8768	7.8819	0.0000	0.1
54	147.2324	0.3067	54.99	16.8673	17.0136	0.0214	0.9
55	143.9846	0.3000	111.9	33.5664	33.6962	0.0168	0.4
56	172.4955	0.3594	64.96	23.3444	23.5719	0.0518	1.0
57	192.6386	0.4013	99.97	40.1210	40.4038	0.0799	0.7
58	109.8553	0.2289	71.03	16.2563	16.6024	0.1198	2.1
59	8.485	0.0177	106.4	1.8808	2.5630	0.4654	26.6
60	13.3441	0.0278	180	5.0040	8.6479	13.2781	42.1
61	10.2386	0.0213	106.4	2.2696	2.2702	0.0000	0.0
62	21.3634	0.0445	180	8.0113	6.2555	3.0827	28.1
63	54.8749	0.1143	54.95	6.2820	6.2393	0.0018	0.7

(Continua alla pagina successiva)

(Continua dalla pagina precedente)

64	5.55E+01	0.1157	55	6.3650	6.6870	0.1036	4.8
65	5.9358	0.0124	74	0.9151	1.4210	0.2560	35.6
66	76.741	0.1599	39.97	6.3903	6.4782	0.0077	1.4
67	160.5315	0.3344	40.98	13.7054	14.0930	0.1503	2.8
68	16.0225	0.0334	180	6.0084	6.5155	0.2571	7.8
69	19.5186	0.0407	140	5.6929	5.4070	0.0817	5.3
70	147.31	0.3069	47.98	14.7249	14.5905	0.0181	0.9
71	93.6041	0.1950	94.95	18.5161	18.2431	0.0745	1.5
72	202.9651	0.4228	65	27.4849	27.2700	0.0461	0.8
73	73.0606	0.1522	61	9.2848	9.5029	0.0476	2.3
74	202.1795	0.4212	65	27.3785	27.3260	0.0028	0.2
75	204.7565	0.4266	69	29.4337	29.5626	0.0166	0.4
76	30.016	0.0625	61	3.8145	3.6974	0.0137	3.2
77	158.5186	0.3302	41.13	13.5831	13.5641	0.0004	0.1
78	39.5366	0.0824	75.65	6.2311	6.3258	0.0090	1.5
79	23.7514	0.0495	101.3	5.0125	5.8472	0.6966	14.3
80	41.0613	0.0855	65.03	5.5630	5.7857	0.0496	3.9
81	42.7163	0.0890	76.99	6.8515	6.8929	0.0017	0.6

Capitolo 5

Caso Studio (parte 2): Sound Event Detection con CRNN

Il presente capitolo è dedicato al secondo approccio utilizzato per rilevare in maniera automatica il tempo di insorgenza (*Onset time*) dei segnali di emissione acustica. Si tratta di una rete neurale convoluzionale ricorrente (CRNN) inserita nel contesto della sound event detection (SED).

5.1 Sound event detection

Con il termine Sound Event Detection (SED) si indica una classe di metodi il cui fine è quello di riconoscere ciò che sta accadendo all'interno di un segnale audio e quando sta accadendo. In altre parole, l'obiettivo è quello di riconoscere in quali istanze temporali diversi suoni sono attivi all'interno di un segnale audio [11].

Figura 5.1. Obiettivo del metodo Sound Event Detection [11]

L'approccio dominante per affrontare tale compito si basa sull'apprendimento supervisionato: un training set di registrazioni audio e le loro annotazioni di riferimento sulle attività di classe sono utilizzati per imparare un modello acustico.
A livello generale, l'attività di rilevamento degli eventi sonori prevede due fasi principali: la rappresentazione delle caratteristiche e la classificazione. Quest'ultima, a sua volta, è composta da due fasi:

1. ***Learning stage***, ovvero una fase di addestramento in cui il sistema apprende la corrispondenza tra le caratteristiche estratte dal segnale audio e un'annotazione che rappresenta l'attività di ogni classe. In altre parole, in questa fase il sistema impara i modelli acustici. Le annotazioni sono rappresentate come una matrice binaria in cui ogni elemento rappresenta una classe attiva (1) o non attiva (0) all'interno di brevi segmenti temporali.
   Poichè i dati di addestramento sono difficili da ottenere a volte, in questa fase viene spesso utilizzato un processo chiamato *data augmentation*. Tale processo permette di aumentare artificialmente la quantità e la diversità dei dati di formazione. Nel caso di segnale contenente singole sorgenti sonore, l'aumento dei dati può essere utilizzato per integrare condizioni acustiche come il rumore o caratteristiche dell'ambiente.

2. ***Test stage***, ovvero una fase in cui il sistema riceve le caratteristiche estratte da una registrazione audio appartenente al test set e fornisce un output rappresentato secondo la stessa convenzione: una matrice che indica l'attività binaria per ogni classe sonora in segmenti di tempo consecutivi. In altre parole, in questa fase i modelli acustici vengono utilizzati per fornire previsioni su nuovi dati.

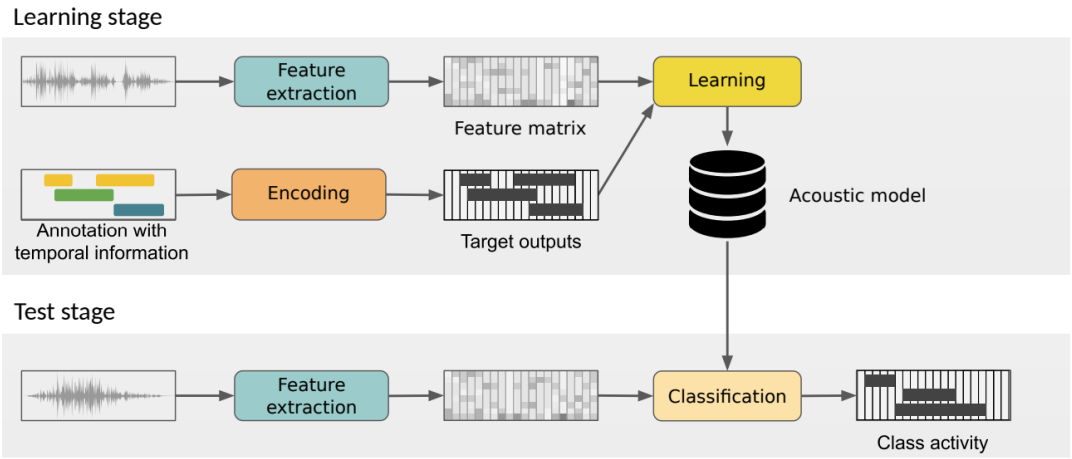

Figura 5.2. Panoramica di un sistema di Sound Event Detection [11]

Dal momento che lo scopo della SED è stimare l'attività temporale dei suoni in una registrazione audio, non necessariamente si applica un'etichetta all'intera registrazione. In questi metodi, è possibile distinguere due tipologie di etichette:

- Etichette forti, o *strong labels*. Esse indicano i segmenti temporali esatti della registrazione audio in cui l'evento è attivo, vale a dire i suoi tempi di inizio e di fine (onset e offset time);

- Etichette deboli, o *weak labels*. Esse informano solo sulla presenza del suono in una registrazione audio, senza indicare esplicitamente la regione temporale in cui questo suono è attivo.

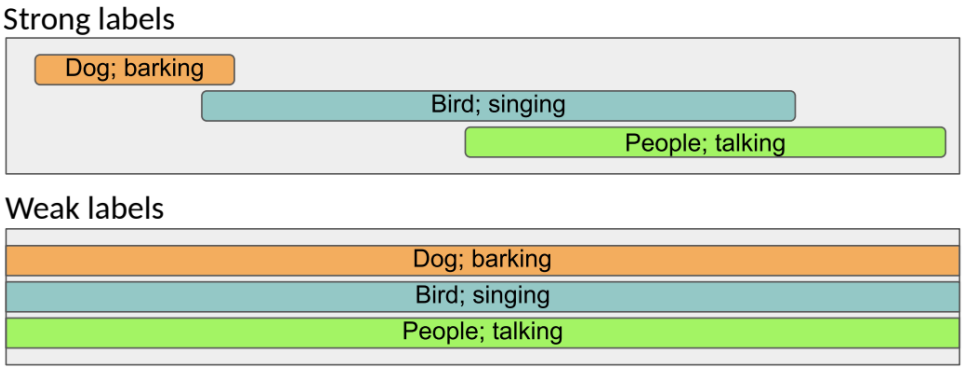

Figura 5.3. Strong e weak labels [11]

Per valutare le prestazioni di un sistema di Sound Event Detection, è possibile utilizzare due tipologie di approccio:

- Valutazione basata sui segmenti, o *segment-based evaluation*. Questo approccio consiste nel confrontare l'output del sistema e l'annotazione di riferimento su una griglia temporale fissa, la quale è più grossolana rispetto alla risoluzione dell'output del sistema. Il confronto si basa sul conteggio degli eventi correttamente ed erroneamente rilevati in termini di veri positivi (TP), veri negativi (TN), falsi positivi (FP) e falsi negativi (FN).

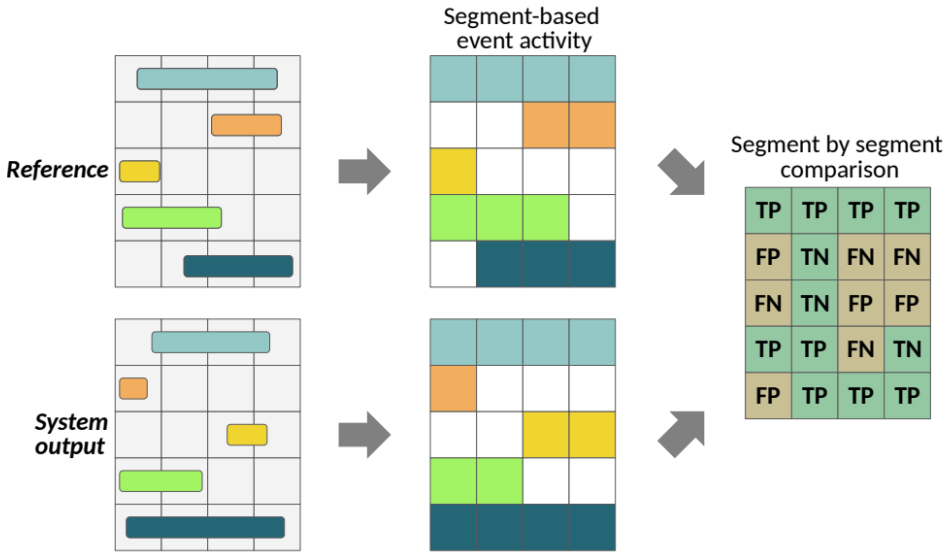

Figura 5.4. Segment-based evaluation [11]

Le metriche comunemente utilizzate nella classificazione dei pattern includono la

precisione (P), il richiamo (R), l'F-score (F) e il tasso di errore (error rate, ER):

$$P = \frac{TP}{TP + FP} \tag{5.1}$$

$$R = \frac{TP}{TP + FN} \tag{5.2}$$

$$F = \frac{2PR}{P + R} \tag{5.3}$$

Mentre P, R ed F-score considerano e contano gli errori singolarmente, ER conta una occorrenza congiunta di un falso positivo e un falso negativo come un singolo errore di sostituzione S. Indicando rispettivamente con I (*insertions*) i falsi positivi e con D (*deletions*) i falsi negativi non contabilizzati in S, ER è calcolato come il numero totale di errori rispetto al numero di eventi di riferimento N:

$$ER = \frac{S + D + I}{N} \tag{5.4}$$

- Valutazione basata su eventi, o *event-based evaluation*. Questo approccio confronta l'output del sistema e l'annotazione di riferimento in termini di istanze sonore, verificando quanto i tempi di onset/offset delle istanze sonore rilevate corrispondano ai tempi delle istanze sonore annotate, evento per evento. La soggettività dei confini temporali viene alleviata consentendo un certo grado di disallineamento, chiamato *collar*, tra la reference e l'output del sistema. Un'istanza di evento è contata come un vero positivo se ha la stessa etichetta dell'evento di riferimento corrispondente e i suoi confini temporali si trovano all'interno del collare temporale consentito rispetto all'evento di riferimento.

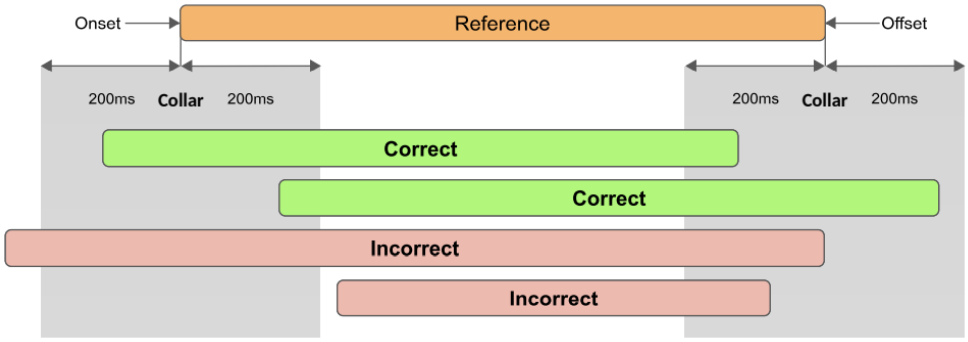

Figura 5.5. Event-based evaluation [11]

Le opzioni di valutazione basate su segmenti e su eventi rappresentano due visioni concettualmente diverse dello stesso output: la valutazione basata su segmenti rappresenta

le prestazioni del sistema nel rilevare le regioni temporali in cui sono attivi eventi sonori, mentre la valutazione basata su eventi rappresenta le prestazioni del sistema nel rilevare singole istanze di eventi sonori.

5.2 Convolutional recurrent neural network (CRNN)

Una rete neurale convoluzionale ricorrente è un'architettura di rete adatta per le attività in cui la modellazione della sequenza temporale è vantaggiosa, come nel caso della SED [11]. Indicata in letteratura con l'acronimo CRNN (convolutional recurrent neural network), essa combina in un'unica architettura le potenzialità di una rete neurale convoluzionale (CNN), di una rete neurale ricorrente (RNN) e di uno strato completamente connesso (FC). L'architettura di rete viene selezionata in base all'attività: gli strati ricorrenti sono inclusi in presenza di sequenze di dati, mentre nei compiti di classificazione, in cui non è necessario preservare le informazioni temporali, le reti di solito includono solo blocchi convoluzionali. Più comunemente, le CRNN contengono dai 2 ai 5 blocchi convoluzionali, mentre il numero di livelli ricorrenti è di solito 1 o 2.

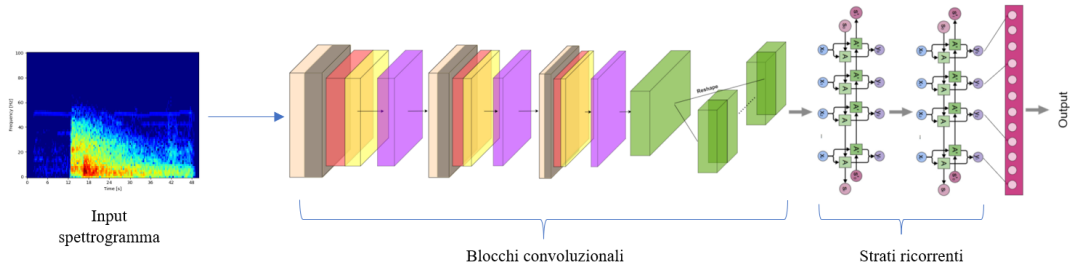

Figura 5.6. Architettura CRNN

La rete inizia con il primo blocco convoluzionale, il quale riceve come input una rappresentazione tempo-frequenza dei dati. Un blocco convoluzionale è costituito da uno strato convoluzionale, una non linearità e uno strato di pooling. Come visto in precedenza, il livello convoluzionale applica un kernel al suo input per creare una nuova rappresentazione delle caratteristiche. Questa rappresentazione passa attraverso uno strato di attivazione non lineare, di solito unità lineari rettificate (ReLU). Infine, lo strato di pooling ha il compito di ridurre la dimensionalità dei dati. Poiché la sound event detection richiede la stima della posizione temporale degli eventi sonori, l'asse temporale deve essere mantenuto: l'operazione di pooling, quindi, viene eseguita solo sull'asse delle frequenze. Seguono poi un numero di blocchi convoluzionali identificati a seconda dell'attività. Gli output dell'ultimo blocco convoluzionale sono sovrapposti sull'asse delle frequenze, dando una rappresentazione 2D che viene poi fornita come input al primo livello ricorrente. Uno strato ricorrente ha il ruolo di apprendere informazioni contestuali dall'evoluzione del segnale nel tempo. Alcune unità specializzate ricorrenti sono bidirezionali, il che significa che tengono conto sia degli input passati che di quelli

futuri, oltre a quello attuale. Infine, in base all'output dell'ultimo livello ricorrente, lo strato FC ha il ruolo di produrre le probabilità di attività degli eventi sonori. Poiché l'output atteso nella SED è una sequenza di indicatori binari di attivazione per ogni classe di evento, l'output di rete risulta essere binarizzato.
In sintesi, dunque, gli strati convoluzionali fungono da estrattori di caratteristiche, con l'obiettivo di apprendere le caratteristiche discriminatorie attraverso le convoluzioni consecutive e le trasformazioni non lineari applicate alla rappresentazione tempo-frequenza presentata all'ingresso della rete. Invece, gli strati ricorrenti mirano ad apprendere le dipendenze temporali nella sequenza di caratteristiche presentate al loro ingresso. Prima di passare ad analizzare il modello di CRNN utilizzato, risulta necessario un breve accenno al concetto di rete neurale ricorrente.

5.2.1 Reti neurali ricorrenti

Una rete neurale ricorrente (recurrent neural network, RNN) è una classe di rete neurale artificiale che include neuroni collegati tra loro a formare dei cicli. In altre parole, i valori di uscita di un layer di livello superiore (più vicino all'uscita) vengono utilizzati come ingresso per un layer di livello inferiore (più vicino all'ingresso). A differenza delle reti feedforward, in cui l'informazione può andare solo in un verso, nelle RNN i loop consentono ai segnali di viaggiare sia in avanti che indietro, creando così un sistema più complesso e meno stabile. Questa caratteristica rende le RNN molto interessanti, in quanto il concetto di ricorrenza introduce intrinsecamente il concetto di memoria di una rete. In una rete RNN, infatti, l'output di un neurone può influenzare se stesso, in uno step temporale successivo, o può influenzare neuroni della catena precedente. Il tipo di dato che tale rete è in grado di trattare è una sequenza temporale o serie temporale.

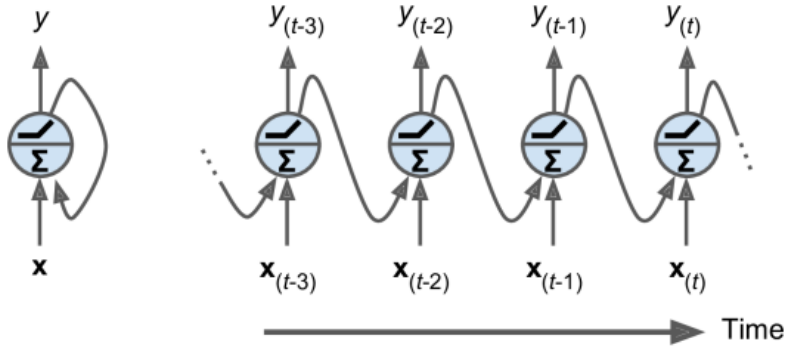

Figura 5.7. Neurone ricorrente (a sinistra) dopo l'esecuzione dell'unfolding in time (a destra) [7]

Si consideri la forma più semplice possibile di RNN, ovvero quella composta da un singolo neurone che riceve un input, producendo un output, e inviando l'output a se stesso. Ad ogni passo temporale t (chiamato anche *frame*), questo neurone riceve

l'input x(t) e l'output y(t-1) dal precedente passo temporale. Ogni neurone ricorrente, dunque, ha due serie di pesi: uno per gli input x(t) e l'altro per le uscite del precedente passo temporale, y(t-1). Chiamiamo questi vettori di peso w_x e w_y. Se consideriamo l'intero strato ricorrente invece di un solo neurone, possiamo posizionare tutti i vettori di peso in due matrici di peso, W_x e W_y. Il vettore di uscita dell'intero strato ricorrente sarà:

$$y_t = \phi\ (W_x^T \cdot x_{(t)} + W_y^T \cdot y_{(t-1)} + b) \tag{5.5}$$

dove $\phi(\cdot)$ rappresenta la funzione di attivazione e b il vettore dei bias.
Dal momento che l'uscita di un neurone ricorrente al passo temporale t è funzione di tutti gli input dei passi temporali precedenti, la rete è in grado di basare le sue decisioni sulla storia passata (effetto memoria). Chiameremo cella di memoria, o più semplicemente *cella*, una parte di rete ricorrente che preserva uno stato interno per ogni istante temporale. Ogni cella è costituita da un numero prefissato di neuroni e può essere considerata come una sorta di livello della rete. In generale, lo stato di una cella al passo temporale t, indicato con h(t) (dove 'h' sta per 'hidden'), dipende dall'input $x_{(t)}$ e dallo stato precedente $h_{(t-1)}$:

$$h_{(t)} = f(h_{(t-1)}, x_{(t)}) \tag{5.6}$$

Per poter addestrare una RNN è necessario eseguire il cosiddetto unfolding o unrolling in time e poi semplicemente usare la backpropagation regolare: questa strategia prende il nome di *backpropagation through time* (BPTT). Con il termine unfolding si indica un'operazione di trasformazione di una rete RNN in una di tipo feedforward. In sostanza, procedere con l'unfolding della rete significa fissare a priori il numero di passi temporali su cui effettuare l'analisi. Di fatto una rete neurale RNN unfolded su 10 step equivale ad una DNN feedforward con 10 livelli. Pertanto addestrare RNN che appaiono relativamente semplici può essere molto costoso e critico per la convergenza (problema della scomparsa/esplosione del gradiente). Quest'ultimo problema può essere alleviato utilizzando varie tecniche, tra le quali il dropout ricorrente o la layer normalization.
Un'altra difficoltà delle RNN, unitamente al problema dei gradienti instabili, consiste nell'avere una memoria a breve termine molto limitata. In altre parole, le celle base hanno difficoltà a ricordare e sfruttare input di step lontani: la memoria dei primi input tende a svanire. Questo limite potrebbe diventare un problema quando si ha la necessità di tener traccia anche di eventi lontani. Se, ad esempio, si considera una frase, diventa fondamentale il contesto delle parole e come le stesse siano tra di loro legate. Ricordare in una sequenza di parole solo quella immediatamente precedente non dà nessun valore aggiunto. Per risolvere questo problema e facilitare la convergenza in applicazioni complesse, sono state proposte celle più evolute dotate di un effetto memoria a lungo termine: tra le più note troviamo LSTM e GRU.

Cella LSTM

La cella *Long Short-Term Memory* (LSTM) è stata proposta nel 1997 da Sepp Hochreiter e Jürgen Schmidhuber [25]. In LSTM lo stato $h_{(t)}$ è suddiviso in due vettori:

- $h_{(t)}$ è uno stato (o memoria) a breve termine;
- $c_{(t)}$, dove 'c' sta per 'cell', è uno stato (o memoria) a lungo termine.

In generale, la cella apprende durante il training cosa è importante dimenticare (*Forget gate*) dello stato passato c(t-1) e cosa estrarre e aggiungere (*Input gate*) dall'input corrente x(t). Per il calcolo dell'output y(t) = h(t) si combina l'input corrente (*Output gate*) con informazioni estratte dalla memoria a lungo termine.

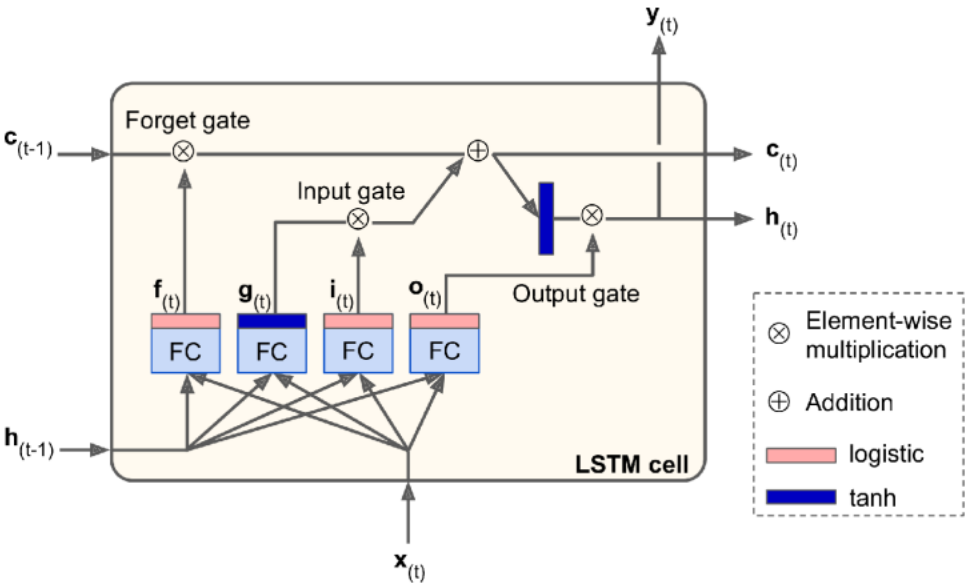

Figura 5.8. Cella LSTM [7]

Più in dettaglio, il vettore di ingresso attuale x(t) e il precedente stato a breve termine h(t-1) sono inviati a quattro diversi strati completamente connessi (FC), i quali hanno tutti uno scopo diverso:

- Il layer principale è quello che restituisce g(t). Il suo ruolo è quello di analizzare gli input attuali x(t) e lo stato precedente a breve termine h(t-1). Esso corrisponde all'unico livello presente in una cella di base, in cui l'uscita va direttamente a y(t) e h(t). Al contrario, in una cella LSTM l'uscita di questo strato viene in parte memorizzata nello stato a lungo termine e in parte dimenticata;
- Gli altri tre strati sono *gate controllers*. Dal momento che usano la funzione di attivazione logistica, le loro uscite vanno da 0 a 1: se emettono 0 chiudono il gate, mentre se producono 1 lo aprono. In particolare:

– Il *Forget Gate* (controllato da f(t)) controlla quali parti dello stato a lungo termine devono essere cancellate;
– L'*Input gate* (controllato da i(t)) controlla quali parti di g(t) devono essere aggiunte allo stato a lungo termine;
– L'*Output gate* (controllato da o(t)) controlla quali parti dello stato a lungo termine devono essere lette e prodotte in questo step temporale, sia in h(t) che in y(t).

Cella GRU

La cella GRU (*Gated Recurrent Unit*) è una versione semplificata di LSTM, che cerca di mantenerne i vantaggi riducendo parametri e complessità. Le principali semplificazioni sono:

- Entrambi i vettori di stato sono fusi in un unico vettore h(t);
- Un singolo gate controller z(t) controlla sia il Forget gate che l'Input gate. Se il gate controller emette un 1, il forget gate è aperto (= 1) e l'input gate è chiuso (1 - 1 = 0). Se emette uno 0, avviene il contrario. In altre parole, ogni volta che uno stato deve essere memorizzato, prima viene cancellata la posizione in cui verrà memorizzato.
- Non è più presente l'Output gate; il vettore di stato completo viene emesso ad ogni passo temporale. Tuttavia, c'è un nuovo gate controller r(t) che controlla quale parte dello stato precedente verrà mostrata allo strato principale (g(t)).

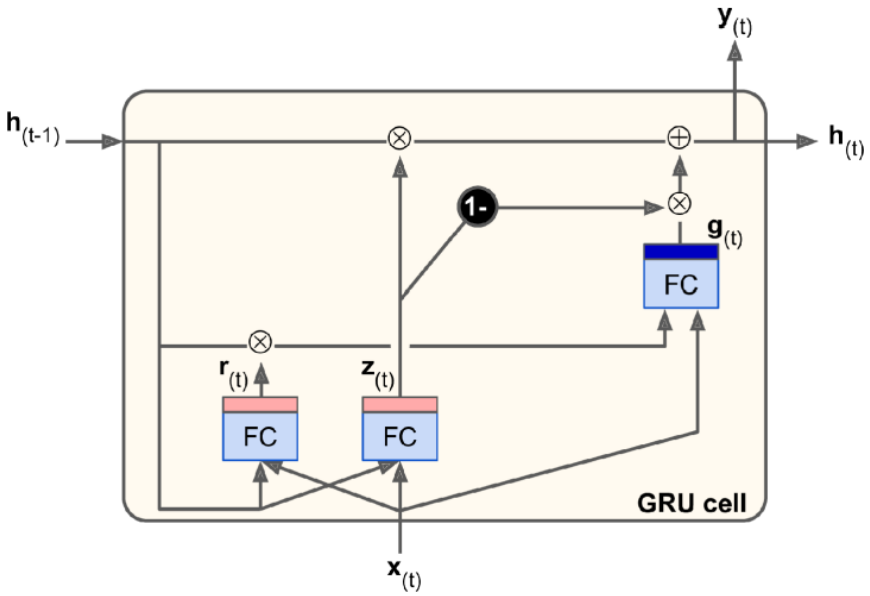

Figura 5.9. Cella GRU [7]

5.3 Analisi del modello utilizzato

Per il rilevamento automatico dell'onset time si è impiegato il metodo della sound event detection, attraverso l'utilizzo di una convolutional recurrent neural network. Il punto di partenza per raggiungere lo scopo sopra citato è stato il codice contenuto all'interno di un Github [34], basato sul metodo proposto nell'articolo '*Sound event detection using spatial features and convolutional recurrent neural network*' [26]. Di seguito vengono illustrati gli step previsti dal modello in esame.

5.3.1 Etichettamento dei dati

Come sottolineato in precedenza, l'approccio dominante per affrontare il compito della sound event detection si basa sull'apprendimento supervisionato. Dunque, il primo passo consiste nell'etichettamento dei dati. Come avvenuto per la precedente rete, la Faster R-CNN, anche in questo caso si parte valutando le prestazioni del modello individuato utilizzando segnali sismici. Dopo aver individuato manualmente il tempo di insorgenza di questi ultimi, l'etichettamento dei dati sarà nella forma:

Nome file	*onset time*	*offset time*	*class label*

Nel caso in esame, considerando un singolo segnale, l'etichettamento risulterà essere:

Nome file	*0*	*onset time*	*Nocrack*
Nome file	*onset time*	*offset time*	*Crack*

Di seguito si riporta la rappresentazione grafica di un segnale etichettato:

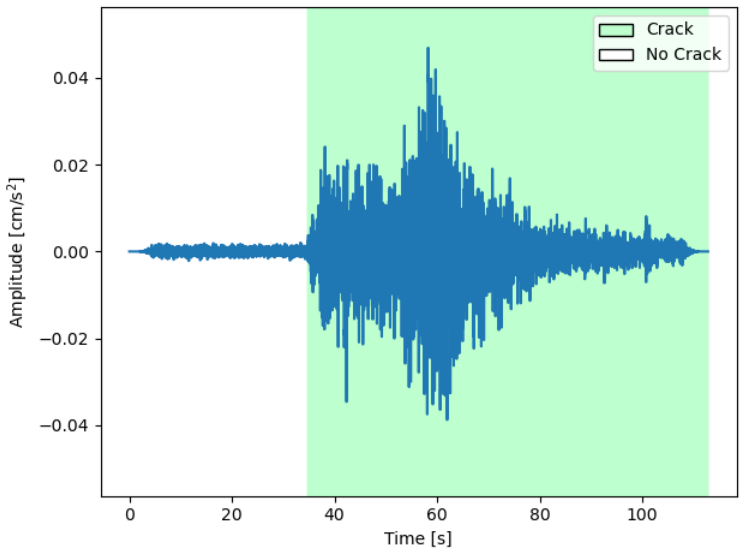

Figura 5.10. Rappresentazione di un segnale con relative etichette

Contemporaneamente all'etichettamento dei dati, avviene il loro splittaggio in training set e test set. Partendo dal medesimo dataset utilizzato nel caso della Faster R-CNN, contenente 410 forme accelerometriche, si utilizzano 328 segnali ($\sim$ 80 % del dataset) per il training set e 82 segnali ($\sim$ 20 % del dataset) per il test set.

5.3.2 Estrazione e rappresentazione delle caratteristiche

Come precedentemente accennato, una rete CRNN riceve come input una rappresentazione tempo-frequenza dei dati. La rappresentazione grafica dell'intensità di un suono in funzione del tempo e della frequenza prende il nome di *spettrogramma*. Esso mostra la relazione tra 3 variabili che caratterizzano qualsiasi suono:

- Frequenza, riportata in ordinata;
- Tempo, rappresentato in ascissa;
- Intensità del suono, rappresentata in scala di grigi o a colori. Se si usa la rappresentazione a colori, i colori sul blu scuro rappresentano un suono di bassa intensità, mentre quelli gialli vengono utilizzati per le alte intensità.

Progettato allo scopo di applicare l'analisi di apprendimento automatico su segnali musicali e audio, il pacchetto Python *Librosa* permette di calcolare in modo efficiente lo spettrogramma per grandi serie temporali in pochi secondi. Librosa memorizza gli spettrogrammi come matrici numeriche bidimensionali, dove il primo asse è la frequenza e il secondo è il tempo.

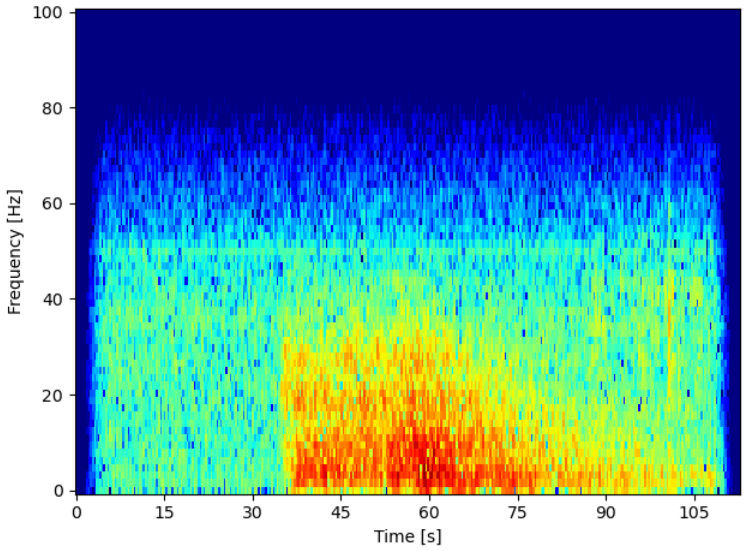

Figura 5.11. Spettrogramma di un segnale sismico

Il secondo passo, dunque, consiste nell'estrarre le etichette, le caratteristiche (ovvero gli spettrogrammi), e normalizzare quest'ultime. Per quanto riguarda le etichette:

- alla class label '*NoCrack*' verrà associato il valore 0;
- alla class label '*Crack*' verrà associato il valore 1;

L'estrazione delle etichette segue la procedura sottostante:

1. si parte dal dizionario contente le informazioni relative all'onset time, alla durata del segnale e alle etichette nella forma:

0	onset time	0
onset time	offset time	1

   e si estraggono le colonne nel seguente modo:
   frame_start =

0
$\frac{onset\ time \cdot sr}{hop\_len}$

   frame_end =

$\frac{onset\ time \cdot sr}{hop\_len}$
$\frac{offset\ time \cdot sr}{hop\_len}$

   dove sr indica la frequenza di campionamento (*sampling rate*) in Hz e hop_len indica il numero di campioni tra frame. Nel caso in esame, sr=200 e hop_len=64. In frame_start si approssimano le varie celle all'intero più piccolo, mentre nel caso di frame_end all'intero più grande. Di seguito viene riportato un esempio:

0	6.0219	0
6.0219	60.04	1

   frame_start =

0
18

   frame_end =

19
188

2. a questo punto viene creata una matrice a due colonne: nella prima colonna viene inserito il numero 1 a partire dal valore contenuto nella prima cella di frame_start (quindi da 0) fino ad arrivare al valore contenuto nella prima cella di frame_end; mentre, nella seconda colonna viene inserito il numero 1 a partire dal valore contenuto nella seconda cella di frame_start fino ad arrivare al valore contenuto nella seconda cella di frame_end. In questo modo, in corrispondenza dell'onset time si troverà una coppia di 1. Continuando l'esempio, si ottiene:

0	1	0
1	1	0
2	1	0
3	1	0
4	1	0
5	1	0
6	1	0
7	1	0
8	1	0
9	1	0
10	1	0
11	1	0
12	1	0
13	1	0
14	1	0
15	1	0
16	1	0
17	1	0
18	1	1
19	0	1
⋮	0	1
184	0	1
185	0	1
186	0	1
187	0	1

Tabella 5.1. Esempio di etichette estratte da un segnale

Risulta evidente come al diminuire di hop_len, aumenti la discretizzazione del segnale e, di conseguenza, la precisione nell'individuazione di onset e offset time.
Il risultato di questa fase, nonchè input della CRNN, è un file (.npz) nella forma:

X_train *Y_Train* *X_Test* *Y_Test*

dove:

1. X_train e X_test contengono rispettivamente la concatenazione delle caratteristiche normalizzate estratte dai segnali appartenenti al training e al test set;

2. Y_Train e Y_Test contengono rispettivamente la concatenazione delle etichette estratte dai segnali appartenenti al training e al test set.

5.3.3 Addestramento e analisi dell'output della rete

A questo punto è possibile procedere con l'addestramento della rete. La rete in esame è costituita da 3 blocchi convoluzionali e 2 livelli ricorrenti. Di seguito si riporta la schematizzazione dell'architettura della rete utilizzata:

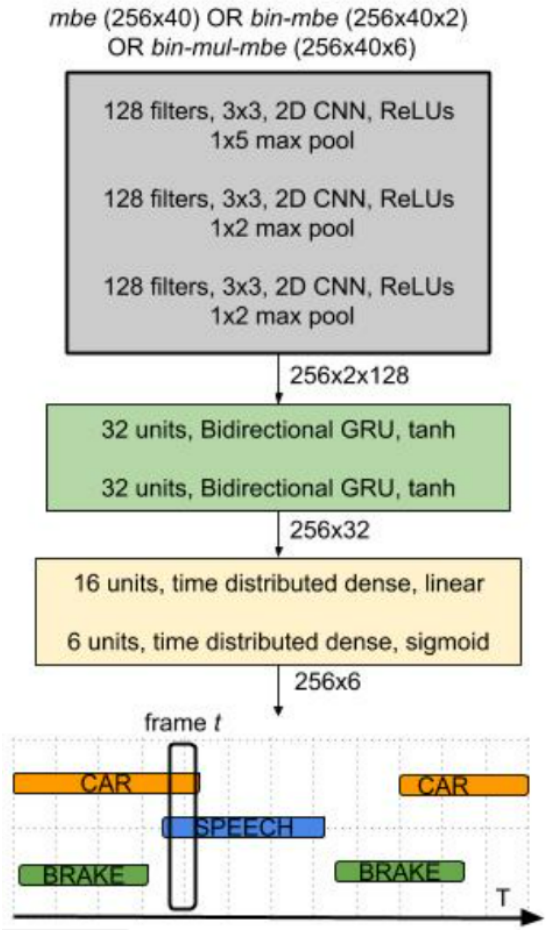

Figura 5.12. Schematizzazione della CRNN utilizzata

In generale, il metodo proposto prende una sequenza di caratteristiche audio in ingresso e predice l'attività delle classi di eventi di destinazione per ciascuno dei fotogrammi di ingresso. In questo caso, come si nota in figura, la lunghezza dei fotogrammi in ingresso è pari a 256. A questo punto, una volta allenata la rete, si procede con l'analisi delle predizioni. La tabella seguente mostra il confronto tra l'annotazione di riferimento (target), l'input e l'output del sistema. Per trasformare la predizione dell'onset time in secondi si utilizza l'inverso della formula precedente, ovvero:

$$onset\ time(s) = \frac{onset\ time_{output} \cdot hop\_len}{sr} \tag{5.7}$$

Tabella 5.2: Confronto tra target, input e output della CRNN

	Target (s)		**Y_Test**			**Predizione_rete**		
N°	onset time	offset time	onset time	offset time	onset time (s)	onset time	offset time	onset time (s)
0	6.6766	76.85	21	241	6.72	22	238	7.04
1	34.3123	112.8	349	594	34.56	349	591	34.56
2	6.6663	76.85	615	835	6.72	615	831	6.72
3	17.6947	43.88	891	973	17.92	890	972	17.6
4	25.7180	69	1053	1189	25.92	1053	1187	25.6
5	17.3917	45.98	1244	1333	17.6	1244	1333	17.6
6	8.0809	33.88	1359	1439	8.32	1358	1439	8
7	21.9424	59.98	1508	1627	22.08	1508	1627	22.08
8	6.2462	80.18	1647	1878	6.4	1647	1877	6.4
9	21.3388	64	1945	2079	21.44	1946	2078	21.76
10	21.2586	77.83	2146	2323	21.44	2146	2321	21.44
11	16.1888	45.98	2374	2467	16.32	2374	2466	16.32
12	21.2394	55.98	2534	2642	21.44	2534	2641	21.44
13	7.2648	52.04	2665	2805	7.36	2665	2805	7.36
14	3.3342	75	2816	3040	3.52	2816	3039	3.52
15	16.1084	69.99	3091	3259	16.32	3091	3258	16.32
16	21.4208	98.31	3326	3567	21.44	3328	3567	22.08
17	19.7650	95.4	3629	3866	19.84	3630	3861	20.16
18	11.4942	66.72	3902	4075	11.52	3902	4074	11.52

(Continua alla pagina successiva)

(Continua dalla pagina precedente)

19	16.5483	85.03	4127	4341	16.64	4128	4338	16.96
20	6.3929	108.3	4361	4680	6.4	4362	4676	6.72
21	6.4869	64.96	4701	4884	6.72	4700	4883	6.4
22	0.7498	46	4887	5028	0.96	4887	5028	0.96
23	18.8218	69.99	5087	5247	18.88	5087	5246	18.88
24	7.2168	65	5270	5451	7.36	5269	5450	7.04
25	17.2736	90	5505	5733	17.28	5505	5723	17.28
26	7.0559	111.6	5756	6083	7.36	5756	6081	7.36
27	26.7501	155.9	6167	6571	26.88	6166	6566	26.56
28	11.7133	105	6608	6900	11.84	6608	6898	11.84
29	34.2534	111.1	7008	7248	34.56	6982	7245	26.24
30	34.5188	111.7	7356	7598	34.56	7356	7594	34.56
31	12.7141	64.97	7638	7802	12.8	7638	7800	12.8
32	13.9195	44.99	7846	7943	14.08	7846	7941	14.08
33	8.7702	31.96	7971	8043	8.96	7971	8044	8.96
34	13.3461	49.99	8085	8200	13.44	8085	8199	13.44
35	23.5893	64.96	8274	8404	23.68	8274	8401	23.68
36	15.9862	62.36	8454	8599	16	8454	8597	16
37	15.3573	53.97	8647	8768	15.36	8647	8768	15.36
38	39.1341	80.74	8890	9021	39.36	8890	9019	39.04
39	7.4119	28	9045	9109	7.68	9023	9109	0.64
40	8.6965	31.96	9137	9209	8.96	9136	9210	8.64

(Continua alla pagina successiva)

(Continua dalla pagina precedente)

41	23.5801	64.96	9283	9413	23.68	9283	9410	23.68
42	40.5800	99.97	9540	9726	40.64	9539	9720	40.32
43	12.1891	54.97	9765	9898	12.48	9764	9896	12.16
44	6.7173	31.95	9919	9998	6.72	9919	9999	6.72
45	12.4685	35.03	10037	10108	12.48	10037	10108	12.48
46	12.4875	64.97	10148	10312	12.8	10147	10311	12.48
47	10.8104	50.04	10346	10469	10.88	10346	10467	10.88
48	7.3759	28	10493	10557	7.68	10491	10557	7.04
49	3.4756	24.04	10568	10633	3.52	10568	10633	3.52
50	27.4446	79.97	10719	10883	27.52	10717	10881	26.88
51	7.5435	32.99	10907	10987	7.68	10906	10987	7.36
52	23.4023	73.45	11061	11217	23.68	11060	11215	23.36
53	7.8819	39.98	11242	11342	8	11242	11343	8
54	17.0136	54.99	11395	11514	17.28	11395	11512	16.96
55	33.6962	111.9	11620	11864	33.92	11618	11860	33.28
56	23.5719	64.96	11938	12068	23.68	11938	12066	23.68
57	40.4038	99.97	12194	12381	40.64	12194	12377	40.32
58	16.6024	71.03	12433	12603	16.64	12433	12602	16.64
59	2.5630	106.4	12612	12936	2.88	12609	12935	1.92
60	8.6479	180	12963	13499	8.96	12959	13485	7.36
61	2.2702	106.4	13507	13832	2.56	13506	13831	2.24
62	6.2555	180	13852	14395	6.4	13852	13481	6.4

(Continua alla pagina successiva)

(Continua dalla pagina precedente)

63	6.2393	54.95	14415	14567	6.4	14415	14566	6.4
64	6.6870	55	14588	14739	6.72	14588	14739	6.72
65	1.4210	74	14744	14971	1.6	14744	14975	1.6
66	6.4782	39.97	14992	15096	6.72	14991	15096	6.4
67	14.0930	40.98	15140	15225	14.4	15140	15525	14.08
68	6.5155	180	15246	15788	6.72	15247	15774	7.04
69	5.4070	140	15805	16226	5.44	15806	16221	5.76
70	14.5905	47.98	16272	16376	14.72	16272	16375	14.72
71	18.2431	94.95	16434	16673	18.56	16434	16670	18.56
72	27.2700	65	16758	16877	27.52	16758	16874	27.2
73	9.5029	61	16907	17068	9.6	16907	17066	9.6
74	27.3260	65	17153	17272	27.52	17153	17269	27.2
75	29.5626	69	17364	17488	29.76	17365	17481	29.76
76	3.6974	61	17500	17679	3.84	17500	17678	3.84
77	13.5641	41.13	17722	17808	13.76	17722	17808	13.76
78	6.3258	75.65	17828	18045	6.4	17828	18044	6.4
79	5.8472	101.3	18064	18362	6.08	18064	18359	6.08
80	5.7857	65.03	18380	18566	6.08	18380	18565	5.76
81	6.8929	76.99	18588	18807	7.04	18588	18688	7.04

(Continua alla pagina successiva)

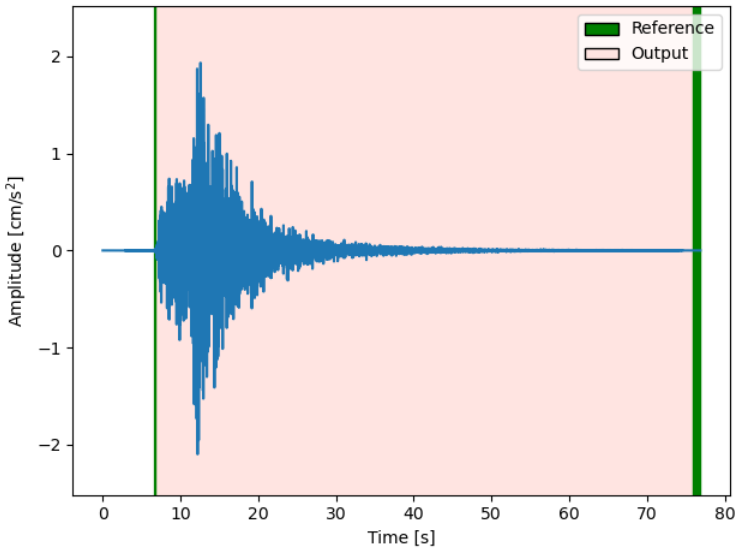

Figura 5.13. Esempio di rappresentazione grafica delle predizioni di CRNN

Come metriche di valutazione delle prestazioni della rete sono state utilizzare l'F-score (vedi 5.3) e l'error rate (vedi 5.4). Considerando che un modello SED ideale ha un error rate pari a 0 e un F-score pari a 1, i valori ottenuti nel caso in esame sono:

	Error rate (ER)	**F-score**
valori ideali	0	1
valori ottenuti	0.012080	0.99197

Tabella 5.3. Valori delle metriche di valutazione SED ottenuti

Capitolo 6

Conclusioni

Il presente lavoro di Tesi ha come obiettivo l'individuazione automatica del tempo di insorgenza (*onset time*) dei segnali di emissione acustica con l'ausilio di algoritmi di intelligenza artificiale (AI), più in particolare di reti neurali artificiali (ANN).
La prima parte dell'elaborato è dedicata ad un'introduzione al vasto mondo dell'intelligenza artificiale, ponendo particolare attenzione al Machine Leaning e al Deep Learning. Seguono poi un'analisi della tecnica di emissione acustica (AE) e un approfondimento delle reti neurali artificiali, oggetto dell'analisi.
La seconda parte dell'elaborato, invece, è dedicata all'analisi delle due metodologie individuate (*Faster R-CNN* e una *CRNN* nell'ambito della *sound event detection*) per il raggiungimento dello scopo sopra indicato e dei risultati ottenuti. Per valutare le prestazioni delle due reti ci si è avvalsi dell'intensa cooperazione che esiste da molti anni tra sismologi terrestri e ingegneri civili. In altre parole, per l'adestramento delle reti è stato utilizzato un dataset composto da segnali sismici dato l'evidente parallelismo tra quest'ultimi e i segnali AE. È importante sottolineare come questa idea sia scaturita dal presupposto che le reti neurali artificiali richiedano la presenza di grandi quantità di dati per realizzare un processo di apprendimento. La realizzazione di un dataset consistente è stata resa possibile grazie a ITACA, l'archivio italiano delle forme d'onda accelerometriche, il quale contiene più di 50.000 forme d'onda.
Sebbene entrambe le reti abbiano permesso il raggiungimento dell'obiettivo prefissato, per la prima è stato necessario introdurre una forzatura nel plottaggio delle forme d'onda per aiutare la rete nell'individuazione. Tra le due reti individuate, dunque, la seconda risulta essere quella più vantaggiosa per il caso in esame. Questo è dovuto al fatto che, oltre ad aver raggiunto valori delle metriche di valutazione molto prossimi ai valori ideali, la rete è in grado anche di identificare la modalità di fessurazione. Infatti, come approfondito nel capitolo 5, la CRNN inserita nell'ambito della sound event detection tratta il problema dell'individuazione del tempo di insorgenza come un problema di classificazione. Se, dunque, durante l'etichettamento dei dati si attribuissero direttamente le etichette '*No Crack*', '*Modo I*' e '*Modo II*', la rete restituirebbe l'onset time e l'offset time dell'evento congiuntamente alla tipologia di fessura. È importante

sottolineare che la discriminante tra le due modalità dipende fortemente dalla tipologia di materiale in cui la fessura si crea e dal tipo di evento.
Il passo successivo al seguente lavoro di Tesi consisterà nell'applicazione delle due reti a segnali di emissioni acustica registrati a seguito di una prova in laboratorio.

Bibliografia

[1] Hadi Salehi and Rigoberto Burgueño. Emerging artificial intelligence methods in structural engineering. *Engineering structures*, 171:170–189, 2018.

[2] Chia-Ming Chang, Tzu-Kang Lin, and Chih-Wei Chang. Applications of neural network models for structural health monitoring based on derived modal properties. *Measurement*, 129:457–470, 2018.

[3] Alberto Carpinteri. *Meccanica dei Materiali e della Frattura*. Pitagora, 1992.

[4] Arash Behnia, Hwa Kian Chai, and Tomoki Shiotani. Advanced structural health monitoring of concrete structures with the aid of acoustic emission. *Construction and Building Materials*, 65:282–302, 2014.

[5] Avik Kumar Das, Deepak Suthar, and Christopher KY Leung. Machine learning based crack mode classification from unlabeled acoustic emission waveform features. *Cement and Concrete Research*, 121:42–57, 2019.

[6] Sebastian Raschka. *Machine learning con Python: costruire algoritmi per generare conoscenza*. Apogeo, 2017.

[7] Aurélien Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems.* " O'Reilly Media, Inc.", 2019.

[8] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[9] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[10] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

[11] Annamaria Mesaros, Toni Heittola, Tuomas Virtanen, and Mark D Plumbley. Sound event detection: A tutorial. *IEEE Signal Processing Magazine*, 38(5):67–83, 2021.

[12] Lu Cheng, Haohui Xin, Roger M Groves, and Milan Veljkovic. Acoustic emission source location using lamb wave propagation simulation and artificial neural network for i-shaped steel girder. *Construction and Building Materials*, 273:121706,

2021.

[13] Alan M Turing. Computing machinery and intelligence. In *Parsing the turing test*, pages 23–65. Springer, 2009.

[14] John R Searle. Minds, brains, and programs. *Behavioral and brain sciences*, 3(3):417–424, 1980.

[15] Douglas R Hofstadter and Daniel C Dennett. *L'io della mente: Fantasie e riflessioni sul sé e sull'anima*. Adelphi Edizioni spa, 2021.

[16] Tom M Mitchell et al. Machine learning. 1997. *Burr Ridge, IL: McGraw Hill*, 45(37):870–877, 1997.

[17] Jose Luis Blanco, Steffen Fuchs, Matthew Parsons, and Maria João Ribeirinho. Artificial intelligence: Construction technology's next frontier. *Building Economist, The*, (Sep 2018):7–13, 2018.

[18] Charles R Farrar, Thomas A Duffey, Scott W Doebling, and David A Nix. A statistical pattern recognition paradigm for vibration-based structural health monitoring. *Structural Health Monitoring*, 2000:764–773, 1999.

[19] A Carpinteri, J Xu, G Lacidogna, and A Manuello. Reliable onset time determination and source location of acoustic emissions in concrete structures. *Cement and concrete composites*, 34(4):529–537, 2012.

[20] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[21] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[22] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013.

[23] Florian Finck, Jochen H Kurz, Christian U Grosse, and Hans-Wolf Reinhardt. Advances in moment tensor inversion for civil engineering. In *International Symposium on Non–Destructive Testing in Civil Engineering*, 2003.

[24] Sumire Kawamoto. Acoustic emission and acousto-ultrasonic techniques for wood and wood-based composites: a review. 2002.

[25] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[26] Sharath Adavanne, Pasi Pertilä, and Tuomas Virtanen. Sound event detection using spatial features and convolutional recurrent neural network. In *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 771–775. IEEE, 2017.

Sitografia

[27] https://www.intelligenzaartificiale.it/.

[28] https://www.tensorflow.org/tutorials/images/transfer_learning.

[29] https://www.image-net.org/index.php/.

[30] https://cocodataset.org/.

[31] https://pytorch.org/.

[32] https://colab.research.google.com/drive/1NziO_b-SW9KmWFh-6C8to9H_QAdpmCBZ?usp=sharing.

[33] http://itaca.mi.ingv.it/ItacaNet_31/#/home.

[34] https://github.com/YashNita/sound-event-detection-winning-method.