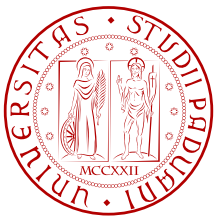


UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Tesi di Laurea Magistrale in Ingegneria Informatica

PROGETTAZIONE E SVILUPPO DI SOFTWARE PER IL RICONOSCIMENTO E LA CLASSIFICAZIONE DI DISCONTINUITÀ NEI NASTRI MAGNETICI AUDIO, BASATO SU TECNICHE DI COMPUTER VISION E NEURAL NETWORK

Laureanda: Ylenia Grava

Relatore: Prof. Sergio Canazza Targon

Correlatore: Ing. Niccolò Pretto

Padova, 1 aprile 2019

Anno Accademico 2018/2019

Sommario

Applicazioni sviluppate *ad hoc* possono offrire un valido supporto agli operatori (tecnici audio e musicologi) nelle procedure di *rimediazione*, agevolandoli nelle procedure più ripetitive e inclini a errori.

Lo scopo della tesi è la ristrutturazione di un software di intelligenza artificiale che identifichi e segnali la presenza di discontinuità sul nastro magnetico, indicandone l'istante temporale e la tipologia.

Dopo un primo approfondimento relativo ai supporti di registrazione e al processo di *rimediazione*, viene presentato il software sviluppato al Centro di Sonologia Computazionale dell'Università di Padova per l'estrazione e la classificazione di frame dai video, acquisiti in fase di digitalizzazione dell'audio mentre il nastro scorre sulle testine del magnetofono.
Dall'analisi di questo software sono stati pianificati, e di seguito implementati, gli interventi per ampliarne le funzionalità e la precisione, applicando tecniche di *computer vision* e reti neurali.

Ringraziamenti

Giunta ormai alla conclusione di questo percorso di studi mi sorge spontanea una riflessione su quanto questo periodo accademico sia stato significativo per me, non solo dal punto di vista formativo ma anche umano. Questi anni mi hanno cambiato profondamente, mi hanno dato soddisfazioni che non avrei mai pensato di ottenere e mi hanno fatto incontrare persone con cui ho trascorso momenti indimenticabili.

Desidero innanzitutto ringraziare il professor Canazza per i preziosi consigli che mi ha elargito durante la stesura di questa tesi, ma soprattutto per la sua cordialità e per l'entusiasmo che ha saputo trasmettermi. Un grazie anche a tutto il laboratorio del CSC, in particolare a Niccolò, che mi ha guidato in questo lavoro con pazienza e dedizione, e a Roberto per la sua disponibilità.

Un ringraziamento speciale ai miei genitori, che hanno sempre creduto in me, appoggiando le mie scelte e spronandomi ogni giorno a dare il meglio.

Voglio inoltre ringraziare tutti gli amici che in questi anni hanno condiviso con me i momenti di spensieratezza e che sono stati sempre presenti nei momenti più difficili, non facendomi mai sentire sola.

In particolare, voglio esprimere un grazie di cuore a Dario per la sua presenza e il suo supporto costanti, ad Andrea che mi ha fatto trascorrere questi anni tra battute e risate e su cui so di poter sempre contare, ad Elena per la sua sincerità e la sua lealtà ed infine a Matteo per gli innumerevoli passaggi in auto e i vent'anni d'amicizia nonostante le nostre personalità così differenti.

Per aspera ad astra

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

Le scienze dure e le discipline umanistiche sono sempre più connesse tra loro e molti campi di ricerca sono interdisciplinari. In particolare, le scienze dell'informazione abilitano innovazioni importanti e all'avanguardia in molti campi di studio. Uno degli ambiti in cui al giorno d'oggi le scienze dell'informazione sono di grande supporto all'arte è il recupero e il mantenimento di opere letterarie, artistiche o culturali, che vanno a formare il cosiddetto patrimonio culturale materiale. Nel corso della storia, però, si è venuto anche a creare l'*intangible cultural heritage* (chiamato anche *memory of the world* [6]), ovvero l'insieme di conoscenze, espressioni e tradizioni orali, che costituiscono il patrimonio culturale di un preciso gruppo sociale in una determinata epoca storica. È essenziale valorizzare e non perdere tale memoria, rilevante tanto quanto il patrimonio materiale. In questo contesto assumono notevole importanza anche i documenti sonori storici, che vanno a formare il *music cultural heritage*. Col passare degli anni, infatti, l'innovazione tecnologica ha permesso all'uomo di usare strumenti di registrazione e riproduzione audio sempre più sofisticati e affidabili, abbandonando di volta in volta i supporti ritenuti obsoleti, dando vita a un vasto e vario patrimonio di documenti sonori di grande interesse storico e scientifico. Basti pensare alle numerose registrazioni di interviste, reportage e cronache, ma anche di concerti o sessioni musicali: una ricca documentazione da tramandare e preservare nel tempo, in quanto testimonianza forte e diretta di epoche ormai passate e oggetto di interesse per molti studiosi, tra cui linguisti e musicologi [8].

Secondo l'UNESCO, tuttavia, tale patrimonio corre il rischio di essere perduto [6]. Tra le molteplici cause si può individuare proprio la continua evoluzione tecnologica; i vecchi supporti su cui vengono effettuate le registrazioni e la strumentazione necessaria a fruire le informazioni ivi contenute sono vittime dell'obsolescenza, con la conseguente cessazione della loro produzione, poiché vengono soppiantati da tecnologie più all'avanguardia. A questo si aggiunge la diminuzione di quantità di pezzi di ricambio disponibili sul mercato, necessari per far fronte a eventuali rotture o malfunzionamenti dei dispositivi di riproduzione[5]. Inoltre, i supporti fisici sono vulnerabili, essendo soggetti a usura e degrado nel tempo, col conseguente rischio di perdere il contenuto registrato.

È pertanto necessario studiare e adottare tecniche sistematiche e *ad hoc* per la salvaguardia e la conservazione nel tempo di tale patrimonio. Definire un protocollo di conservazione efficace è

complesso, vista la grandissima varietà di supporti, materiali, tecniche e standard di registrazione esistenti; per tale motivo adottare un approccio ingegneristico può essere di grande aiuto e di fondamentale importanza, tant'è vero che negli ultimi decenni si è assistito a una rapida crescita dell'applicazione di tecnologie informatiche in supporto alla conservazione del patrimonio musicale. Non è sufficiente, infatti, conservare in condizioni ottimali il supporto originale in modo da rallentarne il processo di degradazione, ma è necessario attuare un trasferimento delle informazioni su supporti più evoluti. Nasce quindi il processo di *rimediazione*, tecnica utilizzata presso il Centro di Sonologia Computazionale (CSC) dell'Università degli Studi di Padova, presentata in 1.2 per l'analisi e l'archiviazione di documenti sonori storici [1].

1.1 Conservazione di documenti sonori storici

Secondo l'Unesco ([14],[11]) i documenti sonori storici possono essere divisi a seconda del tipo di supporto:

- meccanici (e.g. cilindri fonografici)
- magnetici (e.g. esempio bobine, audiocassette)
- ottici (e.g. CD audio)
- digitali (e.g hard disk)

Le diverse tecnologie sono soggette a tipi di usura diversi, a seconda dei materiali di cui sono composte. Tra le cause più comuni e ricorrenti vi sono la scarsa cura nel maneggiare tali supporti (con il rischio di apportare danni meccanici come graffi o distorsioni) e la conservazione in ambienti con condizioni di temperatura sfavorevoli o con livelli di umidità troppo elevati, favorendo la proliferazione di funghi e muffe, o in ambienti esposti a polvere e impurità.

Il processo di degradazione fisica del supporto può essere rallentato tramite tecniche di conservazione, ma non può essere evitato poichè non può essere fermato. È nata quindi l'esigenza di trovare una soluzione per salvaguardare e tramandare il patrimonio di documenti sonori senza che ne vengano pregiudicate la qualità e l'integrità stessa [1].

1.1.1 Conservazione attiva e passiva

Nel corso degli anni si sono formate diverse correnti di pensiero, che hanno portato alla nascita di due metodologie contrapposte: conservazione passiva e attiva [1].

La **conservazione passiva** mira a prevenire e contrastare le principali cause del deterioramento del supporto originale, tramite tecniche diverse a seconda della sua natura, con l'obiettivo di allungarne la vita, e si divide a sua volta in diretta e indiretta. La prima modalità interviene direttamente sul supporto tramite tecniche di restauro per rallentare i processi di deterioramento e permetterne la conservazione nella maniera più integra possibile. La seconda modalità, invece, cerca di salvaguardare le caratteristiche dei supporti prestando particolari attenzioni e accorgimenti per il loro utilizzo e altresì tramite uno stoccaggio in ambienti controllati.

La **conservazione attiva** ha invece come obiettivo la salvaguardia del contenuto dei documenti nel tempo, copiandolo in un supporto diverso da quello originale, secondo la filosofia del "*preserve the content, not the carrier*". Con l'avvento dell'era digitale, è apparsa sempre più chiara l'importanza della digitalizzazione in questo campo. Con digitalizzazione, si indica il processo di conversione di un segnale dal dominio dei valori continui (segnale analogico) a quello dei valori discreti (segnale digitale). Oltre a introdurre un maggiore grado di longevità nella vita del documento, questa tecnica apporta altri vantaggi, tra cui la fruizione di opere prima difficilmente o parzialmente disponibili, grazie alla facilità con cui si può copiare il materiale digitale. Di conseguenza, si rendono accessibili a un pubblico più ampio quei documenti precedentemente impossibili da consultare nella forma originale.
Tale metodologia di conservazione, però, non è immune da aspetti negativi. In primo luogo sono necessarie competenze specifiche per trattare il supporto nella maniera corretta, in modo da permetterne la digitalizzazione, nonchè una standardizzazione delle tecniche per garantire risultati qualitativamente accettabili per l'uso archivistico (in termini di formati, compressioni, frequenze di campionamento, ecc.). In secondo luogo è necessario disporre di tecnologie di *storage* con sufficiente capienza e affidabilità. Inoltre, i nuovi supporti su cui vengono copiati i documenti non sono immuni dall'obsolescenza e, infine, vi è il rischio di esportare il segnale in formati compressi, inficiando l'integrità e l'originalità del documento stesso [5]. Al fine di garantire il più fedelmente possibile l'autenticità, archivisti e operatori devono seguire una serie di linee guida e procedure standardizzate, stilate proprio con l'obiettivo di evitare errori e pervenire a un risultato il più ottimale possibile.

1.2 Il processo di rimediazione

Sarebbe più appropriato distinguere tra digitalizzazione di un audio (intesa come creazione di una copia in formato digitale) e processo di rimediazione, che è finalizzato alla conservazione di un documento sonoro nella sua interezza e che deve seguire una serie di regole e procedure ben definite. In [1] viene descritto il processo di rimediazione attuato presso il Centro di Sonologia

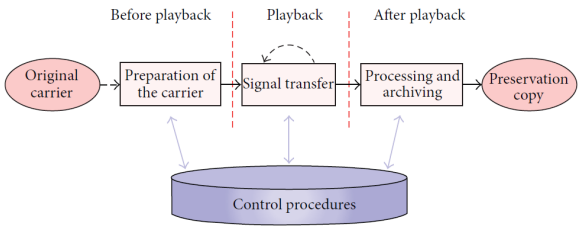


Figura 1.1: *Processo di rimediazione attuato presso il CSC*

Computazionale dell'Università di Padova, il cui schema è riportato in figura 1.1 e che prevede le procedure riportate di seguito.

1 Preparazione del supporto

- 1.1 Documentazione fisica
  - 1.1.1 Documentazione fotografica
  - 1.1.2 Immagini scannerizzate
  - 1.1.3 Convalida dei dati
- 1.2 Ispezione visiva
- 1.3 Analisi chimica
- 1.4 Ottimizzazione del supporto

2 Trasferimento del segnale

- 2.1 Analisi dei parametri e dei formati di registrazione
- 2.2 Configurazione di sistema
  - 2.2.1 Strumentazione per la riproduzione (e.g. magnetofono)
  - 2.2.2 Strumentazione per la *rimediazione* (convertitore, software per l'acquisizione e il monitoraggio, etc.)
- 2.3 Monitoraggio
- 2.4 Convalida dei dati
- 2.5 Archiviazione del supporto originale

3 Elaborazione e archiviazione dei dati

- 3.1 Estrazione dei metadati
- 3.2 Completamente della copia conservativa

Come si può notare dallo schema, la rimediazione non consiste in un semplice riversamento del contenuto del supporto originale in un altro supporto. Infatti, oltre al trasferimento del segnale che deve seguire determinati standard e parametri, vi è un'attenzione particolare al restauro del supporto stesso e alla creazione di una documentazione dettagliata. L'obiettivo è infatti quello di garantire, per quanto possibile, l'integrità e l'originalità del documento nella sua totalità. Pertanto, contestualmente all'acquisizione dell'audio, viene anche creato un set di file multimediali (foto, scannerizzazioni, video) relativi non solo al supporto, ma anche ai documenti cartacei acclusi, come copertine, foto, libretti d'opera. Vengono inoltre opportunamente documentati eventuali "segni di riconoscimento" che caratterizzano il supporto, come per esempio la presenza di giunte di montaggio o di eventuali segni e scritte; si è quindi in grado di ottenere un'affidabilità maggiore sull'originalità del documento.

1.3 Copie conservative

Il risultato del processo di rimediazione è la cosiddetta copia conservativa, ovvero "un insieme organizzato di dati che rappresenta tutta l'informazione portata dal documento originale nella sua complessità, unitamente alla loro descrizione e alla documentazione relativa al processo di conservazione" [3].

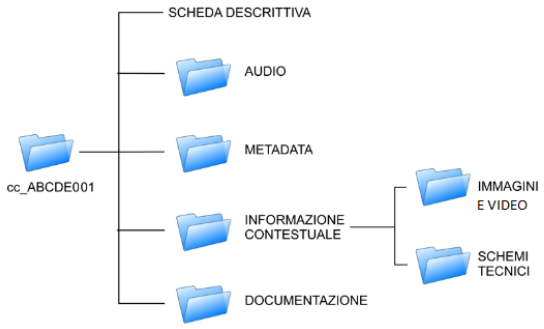


Figura 1.2: *Struttura di una copia conservativa*

La copia conservativa è finalizzata alla conservazione a lungo termine e per questo deve descrivere il documento originale nel minimo dettaglio. La struttura della copia conservativa è presentata in figura 1.2; essa deve contenere gli elementi di seguito riportati.

- File audio, in formato *lossless* BWF (*Broadcast Wave Format*), campionato ad almeno 96kHz e 24bit di risoluzione.
- Documentazione multimediale, composta da foto del supporto originale e video acquisito durante la riproduzione.
- Metadati dei diversi file (*checksum*, specifiche tecniche dei diversi formati).
- Report descrittivo dell'intera copia conservativa e documentazione relativa al processo utilizzato per la rimediazione.

La documentazione video, sincronizzata con l'audio, è di estrema importanza poichè riporta informazioni sullo stato fisico del supporto, sull'eventuale presenza di alterazioni al momento della digitalizzazione. Come vedremo nel paragrafo 1.4.2, il video assume un'importanza primaria quando si tratta di copie conservative di nastri magnetici. Il video, al contrario dell'audio, può essere codificato in formato *lossy*, con l'accorgimento ovviamente di utilizzare risoluzione e bitrate tali da permettere il riconoscimento di alterazioni del supporto durante la riproduzione. Presso il CSC i video vengono acquisiti con una videocamera professionale con una risoluzione di 720x576 pixel.

L'utilizzo di formati e codifiche ad alta qualità, però, aumenta ovviamente le dimensioni della copia conservativa, rendendone difficile la fruizione, in particolar modo se richiesta via web. Inoltre, l'eventuale accessibilità da parte di un pubblico esteso potrebbe mettere a repentaglio

l'integrità stessa del documento. Per questa ragione dalla copia conservativa viene creata la copia d'accesso, che può presentare una qualità sonora inferiore. Questo duplicato, oltre a essere disponibile per la consultazione [9], può permettere al musicologo di testare interventi di restauro o sperimentare analisi di vario genere, impossibili da effettuare sulla copia conservativa senza il rischio di comprometterla.

1.4 Caso di studio: i nastri magnetici

I nastri magnetici per le registrazioni audio furono inventati dal tedesco German Fritz Pfleumer nel 1928, che si basò sulla precedente invenzione di Oberlin Smith e Valdemar Poulsen sviluppata a fine Ottocento: il registratore a filo, ovvero uno strumento in grado di registrare suoni su un filo d'acciaio. I nastri magnetici, che si diffusero negli anni '50 e mantennero la loro supremazia fino agli anni '80, rivoluzionarono il modo di concepire la riproduzione e la trasmissione del suono: per esempio, permisero la registrazione di trasmissioni radio e la conseguente messa in onda in differita [2].
Questo tipo di supporto è fra quelli che presentano più variabili possibili e per questo costituiscono una grande sfida nell'ambito della conservazione.

I nastri magnetici sono costituiti da uno strato ferromagnetico e da un sottostante strato di materiale flessibile, realizzato con acetato, carta o poliestere [13].
Il segnale registrato è prevalentemente analogico, anche se sono stati sviluppati metodi di registrazione digitale (DAT, Digital Audio Tape) [9]. Per quanto riguarda i nastri analogici, le operazioni di scrittura avvengono tramite una testina di incisione, che crea un campo magnetico sul nastro grazie ai segnali elettrici provenienti da una fonte, come ad esempio un microfono; la magnetizzazione del nastro permette di rendere permanente nel tempo l'informazione, codificata sotto forma di campi magnetici. In fase di lettura la testina legge questi campi magnetici traducendoli in segnali elettrici. L'informazione sui nastri digitali, invece, è un segnale discreto; il segnale analogico viene infatti campionato per poi essere scritto sul nastro come flusso di bit. Le operazioni di registrazione e lettura avvengono tramite una testina rotante: essa si muove ruotando contestualmente al nastro che avanza. Essendo la testina inclinata trasversalmente rispetto al nastro, a ogni rotazione la testina scrive su una nuova sezione dello stesso, producendo un campo magnetico che varia a seconda delle informazioni da registrare. Tale tecnica è nota come scansione elicoidale [7].

I nastri magnetici hanno una corta aspettativa di vita; le cause più ricorrenti del loro deterioramento sono riportate di seguito [1].

- **Umidità**: l'acqua a contatto con la superficie del supporto provoca una reazione chimica chiamata idrolisi, che danneggia gravemente il supporto alterandone la composizione chimica e compromettendone l'uso; favorisce inoltre la proliferazione di funghi e muffe.

- **Temperatura**: gli ambienti soggetti a grandi sbalzi di temperatura possono deformare i supporti e ne stressano la composizione chimica; temperature elevate possono infatti favorire e accelerare reazioni come l'idrolisi.

- **Polvere e impurità**: la presenza di agenti esterni può causare rumore durante la riproduzione, poiché rappresentano fisicamente un ostacolo per eventuali testine di lettura, che non riescono ad avere un contatto preciso con la superficie del supporto. Inoltre, eventuali impronte digitali presenti sul supporto diventano un ideale adesivo e aumentano la probabilità che tali impurità si depositino; è pertanto necessario maneggiare i nastri con attenzione.

- **Campi magnetici**: la presenza di tali campi, generati per esempio da attrezzature come i microfoni, può interferire e compromettere anche in modo irreversibile i supporti magnetici.

- **Danni meccanici**: se non maneggiati con cura i nastri magnetici sono soggetti a danni fisici, come distorsioni o stropicciamenti. Tali danni inficiano la riproduzione solo localmente nel punto rovinato.

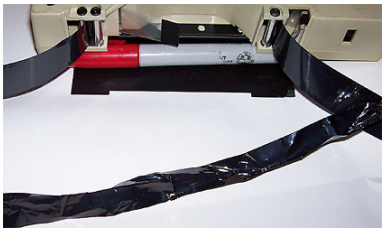


Figura 1.3: *Nastro magnetico corrotto*

I nastri magnetici sono tipicamente raccolti all'interno di bobine o cassette. Nel primo caso, in fase di riproduzione, la bobina che contiene il nastro viene montata su un mozzo (*hub*), mentre l'estremità libera del nastro viene fatta passare attraverso le testine di registrazione e i meccanismi di trascinamento del nastro del dispositivo di riproduzione (chiamato magnetofono, fig 1.4) e infine agganciata a un'altra bobina uguale ma vuota. La bobina può presentarsi in formati e standard diversi: può essere *pancake* (ovvero senza flangia) o meno, può variare per il tipo di *hub*, può possedere flangia di diametri diversi.

Anche il nastro magnetico si può presentare in diverse configurazioni. La larghezza del nastro è variabile:

- nastri da 2 pollici, per utilizzi professionali;

- nastri da 1 pollice;

- nastri da 1/2 pollice;

- nastri da 1/4 di pollice, il formato più diffuso;

- nastri da 1/8 di pollice, per le audio cassette.

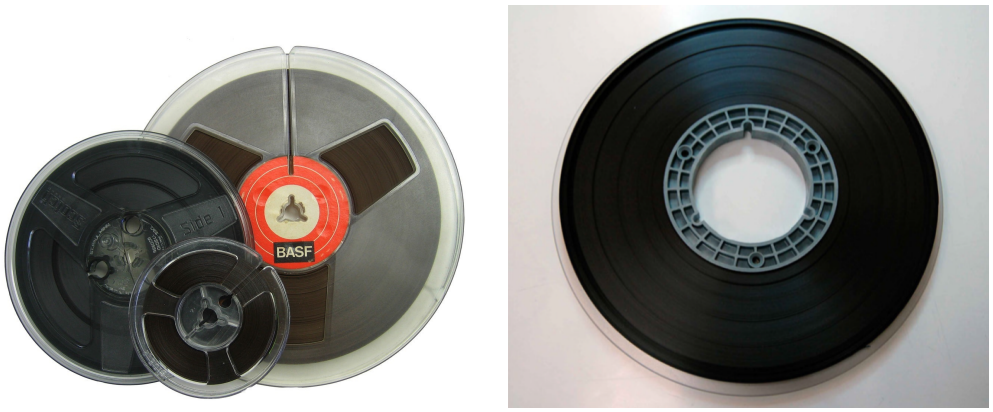
Figura 1.4: *Magnetofono*

A seconda della larghezza del nastro, il numero di tracce, ovvero le suddivisioni longitudinali del nastro, possono variare di numero. Tipicamente il nastro supporta due o quattro tracce, ma esistono casi anche di nastri con 8, 16 o 24 tracce. In ognuna di queste può essere registrato un segnale audio non correlato al segnale audio presente nelle altre piste. Il nastro a due tracce contiene tipicamente un segnale stereo, quello a quattro, invece, può contenere due segnali stereofonici oppure quattro segnali mono indipendenti (quattro tracce registrate nello stesso senso, ottenendo quindi la quadrifonia). Tali configurazioni non seguivano un particolare standard, ma venivano scelte da chi registrava la bobina a seconda delle proprie esigenze. Il nastro può essere magnetico o leader: il primo è quello in cui viene effettivamente registrato il suono dell'opera, mentre il secondo viene utilizzato alle estremità del nastro stesso per essere agganciato alla bobina e non è magnetizzabile (non vi si può registrare nulla e a livello sonoro si percepisce una pausa). I due nastri sono collegati tra loro tramite una giunta, ovvero un'unione realizzata tramite un nastro adesivo (fig. 1.6).

In fase di acquisizione dell'audio per la digitalizzazione dei nastri magnetici è necessario prestare attenzione a come vengono settati alcuni parametri di lettura, per non inficiare la qualità della copia conservativa.

- curva di equalizzazione: serve a filtrare il segnale audio in frequenza. L'equalizzazione è un valore molto importante in fase di riversamento: la scelta di una curva errata distorce il segnale, producendo un risultato infedele al segnale originale e non facilmente riconducibile a quello corretto.

- riduzione del rumore: i segnali sono normalmente affetti da rumori introdotti da varie fonti, non ultimi i sistemi di registrazione stessi. Esistono delle tecniche per ridurre, per quanto possibile, tali disturbi.



(a) **(b)**

Figura 1.5: *Bobina con flangia (a) e* pancake *(b)*

Figura 1.6: *Esempio di giunta con scritta*

- velocità: la velocità di scorrimento del nastro incide sulla qualità della registrazione. A una velocità bassa, infatti, corrisponde una limitazione nella gamma di frequenze correttamente registrabili. I nastri magnetici vengono registrati normalmente a tre differenti velocità: 7.5 ips, 15 ips o 30 ips.

1.4.1 Tape music

Con *tape music* si indica un genere musicale, diffusosi negli anni 50 del Novecento, che ha rivoluzionato il processo di creazione della musica, diventando uno dei fenomeni culturali più importanti del periodo [14]. In Italia, tra i nomi dei compositori più famosi, troviamo Bruno Maderna, Luciano Berio e Luigi Nono. Tale genere musicale è altamente inconvenzionale: il testo musicale è inesistente, non vi è partitura, ma la composizione nasce dalla manipolazione di frammenti e campioni di musica pre-registrata, coinvolgendo fisicamente nella creazione dell'opera il nastro magnetico. Questo infatti viene alterato, tagliato in diversi frammenti, che possono essere sovrapposti in maniera differente in un momento successivo. In questo contesto il compositore

diventa anche colui che performa e il supporto originale dell'opera diventa un reperto unico, non caratterizzabile esclusivamente dal contenuto audio, ma anche dal nastro stesso [2].

Nell'ambito della conservazione, lo scenario della tape music non è di facile gestione, data la quantità di elementi da gestire non riscontrabili in altri supporti. In particolare [2],

- presenza di segni di manipolazione fisica dei nastri, tramite tagli e giunte;
- presenza di annotazioni scritte a mano direttamente sul retro del nastro, essenziali per la riproduzione dell'opera (per esempio la sincronizzazione con un'eventuale orchestra, il formato dell'audio, o i parametri corretti da settare per *velocità* ed *equalizzazione*);
- difficoltà nel settare in modo corretto la velocità di scorrimento del nastro e l'equalizzazione durante la fase di digitalizzazione, in mancanza di informazioni spetta al tecnico audio decidere come settare tali parametri;
- difficoltà nel distinguere tra alterazioni intenzionali e corruzioni dell'audio.

1.4.2 Discontinuità del nastro

Come già detto in 1.4, i nastri magnetici sono estremamente soggetti a degradazioni fisiche. Spesso queste sono riscontrabili a una prima ispezione visiva del supporto (fig. 1.7) e devono essere opportunamente documentate tramite fotografie, che andranno a far parte della copia conservativa.

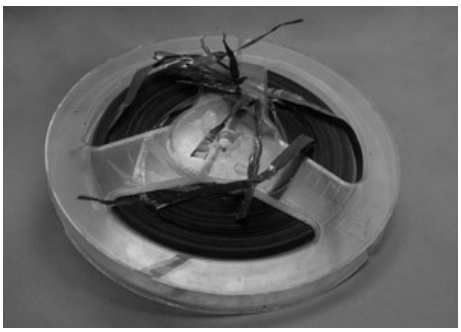


Figura 1.7: *Bobina con nastro magnetico rovinato*

Altri segni o alterazioni del nastro (altrimenti chiamate discontinuità) possono riguardare una zona ristretta e sono riscontrabili solo con la riproduzione dell'intero nastro, quindi in fase di digitalizzazione. In questo contesto il video acquisito durante tale fase è di estrema importanza: i musicologi possono rilevare la presenza di discontinuità visionandolo e di conseguenza intervenire per limitare l'alterazione e rieseguire la digitalizzazione dell'audio da salvare nella copia conservativa.

Le principali discontinuità riscontrabili durante la riproduzione del nastro sono riportate nel seguente elenco.

- Presenza di **danni o rotture del nastro**, che si verificano quando i bordi non appaiono perfettamente dritti o piatti.

- Presenza di **increspature**, conosciute come Kink o Wrinkle [15], ovvero una o più pieghe nel nastro.

- Presenza di **impurità**, come sporco, muffa, polvere o cristalli. Possono anche essere presenti depositi gommosi o sostanze collose sul nastro, che si accumulano sulle testine e sulle guide del magnetofono durante la riproduzione.

- Presenza di **giunte** e/o **segni o scritte** sul nastro o sulle giunte stesse.

- **Aderenza tra gli strati**; consiste nell'adesione della superficie di uno strato alla parte posteriore dello strato successivo, che potrebbe essere causa di distorsioni come *wow* (percepito come una fluttuazione dell'altezza del suono) e *flutter* (conferisce un carattere aspro al suono riprodotto).

- **Perdita del rivestimento magnetico**; le particelle del *backcoat* possono staccarsi dal substrato e accumularsi sul retro del nastro o depositarsi sulle testine del magnetofono. Tali impurità possono compromettere la qualità della riproduzione, poichè impediscono alle testine la corretta lettura.

- **Cupping**; fenomeno per cui il bordo del nastro non è piatto rispetto al centro e si innalza sopra la testina durante la riproduzione.

Sia la creazione che lo studio della copia conservativa condividono un problema comune: l'errore umano. In particolare, visionare il video col nastro che scorre, per lo più tutto uguale, per rilevare la possibile presenza di discontinuità, è un lavoro che richiede una costante attenzione. Dopo ore, il tecnico o musicologo potrebbe non notare alcuni dettagli e il lavoro finale potrebbe presentare alcune lacune. Un paragone è l'opera dei monaci amanuensi, che spesso era affetta da errori dovuti alle ben studiate (in campo filologico) patologie dell'attenzione. Per tale motivo, presso il Centro di Sonologia Computazionale si è deciso di estrarre tali discontinuità in maniera automatica dal video acquisito durante la digitalizzazione dell'audio, sollevando i tecnici audio e i musicologi da un lavoro ripetitivo, stancante e che induce facilmente a commettere errori [15].

Capitolo 2

Stato dell'arte

Oltre al già nominato software per l'estrazione e la classificazione automatica di discontinuità sul nastro magnetico, la cui prima versione verrà presentata in 2.2 e 2.3 e la cui revisione è oggetto di questa tesi, sono state pensate diverse soluzioni per facilitare sia il lavoro dei musicologi sia l'accesso ai documenti da parte di un pubblico ampio.

2.1 Lavori associati

Come spiegato nel capitolo precedente, dalla copia conservativa viene creata la copia d'accesso, finalizzata alla fruizione e all'analisi musicologica; anche quest'ultima deve essere totalmente fedele al documento autentico, per rendere l'esperienza di accesso il più conforme possibile alla consultazione dell'originale.

Nei prossimi due paragrafi verranno presentate alcune delle applicazioni realizzate per rendere possibile quanto sopra descritto.

INTERFACCIA D'ACCESSO

È in via di sviluppo presso i laboratori del CSC un'interfaccia dedicata appositamente all'analisi musicologica dei documenti sonori, presentata in [4]. La vista principale (figura 2.1) permette di caricare tutte le informazioni relative al documento scelto e fornisce la logica per gestire un'analisi musicologica. I diversi elementi sono disposti sull'interfaccia come segue.

- Forma d'onda: nella parte superiore viene mostrata la forma d'onda della traccia; cliccando su un punto qualsiasi la riproduzione verrà portata all'istante selezionato.
- Video: di fianco alla forma d'onda viene visualizzata la documentazione video registrata durante il riversamento.
- Lista di unità: sulla sinistra è presente un elenco delle unità in cui viene divisa l'analisi.
- Lista di attributi: posta al centro, mostra nel dettaglio gli attributi dell'unità corrente.

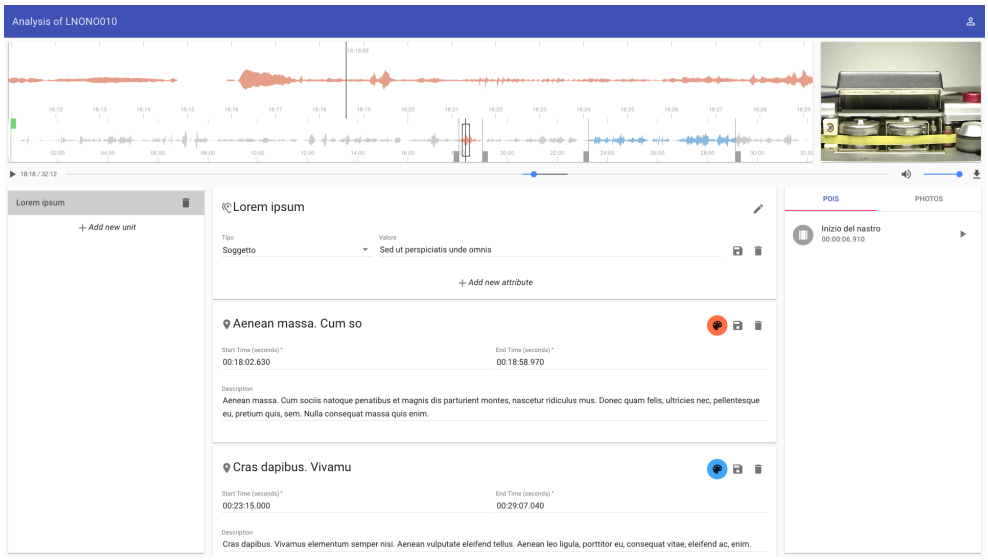


Figura 2.1: *Interfaccia d'accesso*

- Lista di eventi: situata sotto la lista di attributi, mostra quanto avviene in un intervallo di tempo all'interno della traccia in analisi.
- Punti di interesse (POI): scheda posta sulla destra che mostra un elenco dei punti di interesse, sia audio sia video (per esempio la fine del nastro o tutte le discontinuità riscontrabili durante la riproduzione del nastro magnetico).
- Foto: scheda adiacente a quella dei punti d'interesse, che mostra un elenco di foto associate al documento.

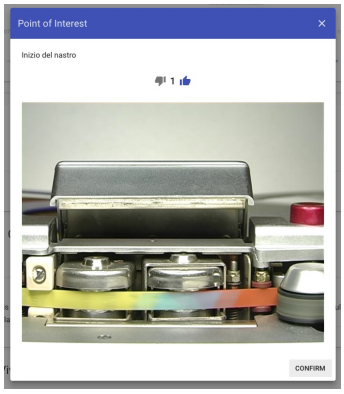


Figura 2.2: *Dettaglio di un POI Video*

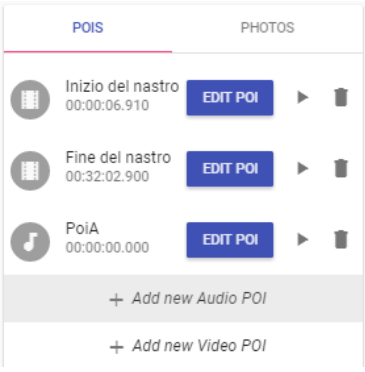


Figura 2.3: *Scheda per la gestione dei POI*

È stata inoltre introdotta la possibilità di inserire, modificare o eliminare manualmente i punti di interesse, come mostrato in figura 2.3, indicandone il tipo (audio o video), il timestamp e il

nome. Tali punti di interesse possono essere valutati dall'utente come rilevanti o meno (figura 2.2): in previsione di un'estrazione automatica dei POI direttamente dal video tale sistema è utile per capire se la discontinuità rilevata è effettivamente da notificare o meno.

REMIND

Per rendere l'esperienza d'accesso il più vicina possibile alla riproduzione fisica del supporto originale è importante attivarsi anche per la conservazione dei sistemi di registrazione e riproduzione. In particolare, la conservazione attiva dei dispositivi, tramite la loro virtualizzazione, permette l'accesso ai documenti in ampia scala. Per questo motivo presso il CSC è stata sviluppata un'applicazione, nata per piattaforme mobili (REMIND) (figura 2.4) ma poi resa disponibile anche via web (REWIND), che simula l'interazione col magnetofono.

Figura 2.4: *Interfaccia di REMIND*

2.2 Rilevamento discontinuità

In [12] viene presentato un primo software per l'elaborazione dei video e l'estrazione automatica dei frame ritenuti interessanti. Sviluppato in C++, con l'utilizzo della libreria OpenCV, l'applicativo analizza un intero video, estrapolando ogni singolo fotogramma ed elaborandolo per determinare la presenza o meno di una discontinuità.

Poiché, per tale lavoro, la zona interessante del video è il nastro, l'algoritmo cerca di rilevare in maniera automatica la sua posizione all'interno del frame. In un primo step il software individua approssimatamente la zona di interesse basandosi sul fatto che il nastro si trova sempre a circa due terzi dell'immagine dal margine superiore; in seguito cerca di affinare la stima rilevando alcune discontinuità e calcolando i *bound* del rettangolo che la circondano: per le ascisse si è calcolata la media dei valori rilevati, mentre per le ordinate si è preso il limite superiore e quello inferiore.

Una volta individuata la zona del nastro, l'algoritmo va ad analizzare e comparare i frame a due a due consecutivi nell'area selezionata dai *bound*. Confrontando i tre canali RGB, calcola il numero di pixel diversi; se tale numero supera una certa soglia (definita come percentuale rispetto alla grandezza del rettangolo in analisi), allora il frame è considerato interessante e viene

salvato, altrimenti tralasciato. Poichè la videocamera utilizzata registra in standard PAL (*Phase Alternating Line*), l'immagine estrapolata è soggetta al fenomeno dell'interlacciamento, per il quale ogni fotogramma viene suddiviso in due semiquadri, formati dalle righe dispari e da quelle pari. Questa tecnica permette all'occhio umano di percepire una qualità di visualizzazione migliore senza bisogno di aumentare la larghezza di banda della trasmissione. Di conseguenza però, al momento dell'estrapolazione del frame, le righe pari e quelle dispari degli oggetti in movimento non risultano allineate (come visibile dalla giunta che scorre in figura 2.5); tale problema è stato risolto salvando solo il semiquadro formato dalle righe dispari, ottenendo quindi immagini con una risoluzione 720x228.

Figura 2.5: *Esempio di immagine interlacciata*

2.2.1 Problematiche riscontrate

L'algoritmo presentato riesce a rilevare buona parte delle discontinuità, ma sono state riscontrate alcune problematiche, sulle quali si è deciso di intervenire (come spiegato in sez. 3.2).

Bound che delimitano l'area di interesse

In primo luogo, controllando i *bound* che l'algoritmo determina automaticamente per rilevare l'area del nastro su cui effettuare l'analisi, si è riscontrato che in molti video essi non sono corretti. È pertanto necessario aggiustarli manualmente, modificando le coordinate dell'ordinata direttamente nel file XML di configurazione (codice 2.1). Tale operazione aumenta notevolmente il lavoro dell'utente e si rivela essere un procedimento molto scomodo, poco preciso e facilmente soggetto a errori: bisogna infatti effettuare dei tentativi cambiando le coordinate fino a quando il rettangolo che individua l'area da analizzare è posizionato correttamente sul nastro, sia come dimensioni che come localizzazione (fig. 2.6). L'altezza del rettangolo infatti non dipende solamente dalla larghezza del nastro, ma anche dal posizionamento della videocamera,

che non è fissa rispetto al magnetofono; pertanto non è detto che la stessa altezza del rettangolo sia riutilizzabile per diversi video.

```
<config>
        <y_u>420</y_u>
        <y_d>472</y_d>
        <x_r>450</x_r>
        <x_l>370</x_l>
        <threshold_percentual>420</threshold_percentual>
</config>
```

Codice 2.1: *File di configurazione*

Figura 2.6: *Rettangolo correttamente posizionato sul nastro*

Numero frame estratti per discontinuità

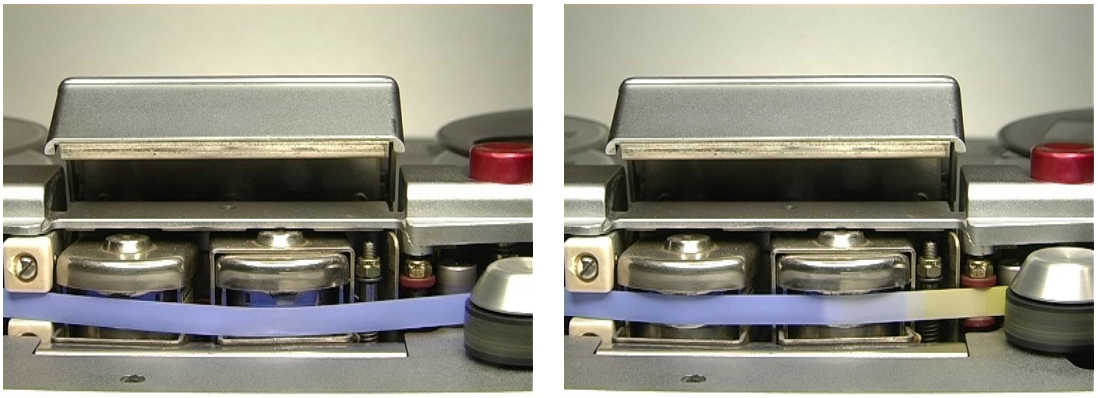
In secondo luogo, per ogni discontinuità che passa sul nastro, l'algoritmo salva in media 3 immagini (quando è in ingresso al frame, in posizione centrale e in uscita dal frame), arrivando però anche a 5 in caso di giunte particolarmente lunghe o presentanti scritte, aumentando notevolmente le dimensioni del dataset risultante.

Presenza di marche

In presenza di marche, rappresentanti il produttore, o in generale di stampe ripetute sul retro del nastro, l'algoritmo distingue ogni singolo frame come differente dal precedente. Quindi tali stampe vengono rilevate continuamente come qualcosa di particolare che deve essere notificato, generando un numero elevatissimo di immagini che di fatto non sono utili allo scopo del progetto. Anche modificando la soglia per discriminare i frame interessanti non si risolve il problema, in quanto si rischia di perdere frame con discontinuità rilevanti; ovviamente è preferibile salvare i falsi positivi (immagini che non risultano essere utili) piuttosto che avere falsi negativi, ovvero perdere informazioni importanti.

Precisione dell'algoritmo

Infine, il software non rileva tutte le discontinuità; una delle problematiche più importanti è il fatto che spesso non rilevi l'evento "fine del nastro". In particolare, questo punto di interesse viene rilevato solo nel caso in cui, nel momento in cui il nastro smette di scorrere tra le testine, si distacchi notevolmente dalla testina di lettura (figura 2.7a)



(a) *Fine nastro* **(b)** *Giunta su nastro che scorre*

Figura 2.7: *Confronto tra l'evento "fine nastro" e il nastro che scorre regolarmente*

Risultati dell'algoritmo

Nella tabella 2.1 vengono riportati i risultati relativi a due video acquisiti filmando lo stesso nastro, che è stato creato *ad hoc* per testare l'algoritmo. Dapprima è stato acquisito il video del nastro a una velocità di 15 ips (Esperimento5.mov) e successivamente a una velocità di 7.5 ips (Esperimento6.mov). Come possiamo notare, l'algoritmo salva moltissimi frame in più rispetto al numero di discontinuità individuate, poiché rileva numerose immagini relative alla stessa giunta e/o scritta che passa. L'algoritmo infine non rileva, per entrambe le velocità, due scritte e la fine del nastro.

Video	Frame salvati	Discontinuità trovate	Discontinuità presenti
Esperimento5	205	76	79
Esperimento6	283	76	79

Tabella 2.1: *Risultati dell'algoritmo di estrazione*

2.3 Classificazione discontinuità

Per la classificazione di tali frame è stato scelto un approccio *deep learning*, utilizzando le reti neurali convoluzionali, tipicamente impiegate per la classificazione di immagini.

Con il software appena presentato sono state quindi estratte le discontinuità e, a partire da video acquisiti presso il CSC, sono stati creati due dataset distinti a seconda della velocità del nastro (7.5 ips e 15 ips), per l'allenamento e il test della rete.

È stata quindi implementata la rete GoogLeNet, proposta in [16], con l'uso del framework Caffe: essendo una rete già pre-allenata su immagini, è sufficiente mantenere gli strati (*layer*) intermedi, modificando alcuni parametri per l'allenamento della rete sul dataset specifico (procedura di *fine-tuning*) e modificando l'ultimo *layer* (per esempio, indicando il numero di classi relative al nostro problema).

Per un primo esperimento, le classi individuate sono state le seguenti.

- `giunte magnetico-magnetico`: collegamenti tra segmenti di nastro magnetico;
- `giunte leader-magnetico`: collegamenti tra nastro di tipo leader e nastro di tipo magnetico; tali giunte non sono sempre dello stesso tipo ma variano in base all'operatore che le crea;
- `marche`: stampe sul nastro che rappresentano il produttore;
- `rovinato`: possibili danneggiamenti e corruzioni della superficie del nastro o deterioramenti dell'ossido;
- `segni`: possibili scritte sul nastro, segni a penna, oppure giunte con la presenza di scritte;
- `ombre`: possibile presenza di zone più scure, che possono essere causate dalla variazione delle luci nella stanza;
- `ondulazioni`: presenza di pieghe sul nastro;
- `sporco`: irregolarità che non rappresentano nastro danneggiato o segni.

Nella tabella 2.2 sono riportati il numero di elementi ottenuti per ciascuna classe per i due dataset.
Come si può notare, le classi `sporco` e `ondulazioni` contengono troppi pochi elementi per poter eseguire un *training* adeguato e pertanto sono state rimosse dagli esperimenti.

Dalle analisi effettuate è emerso che la rete fatica a distinguere tra giunta magnetico-magnetico e leader-magnetico. Tale difficoltà è dovuta al fatto che geometricamente non sono diverse e, in

Classe	Dataset 7.5	Dataset 15
Giunte leader-magnetico	1.038	1.457
Giunte magnetico-magnetico	210	555
Fine nastro	2.484	85
Marche	5.834	5.709
Rovinato	444	175
Ondulazioni	6	0
Ombre	1.333	51
Segni	118	346
Sporco	8	9

Tabella 2.2: *Numero di elementi per ogni classe nei dataset creati*

molte situazioni, è complicata la distinzione anche eseguendo l'operazione manualmente. Se si uniscono le due classi in `giunte`, le *accuracy* migliorano notevolmente.

Inoltre, la rete rileva difficilmente dove il nastro è rovinato e dove sono presenti dei segni: il classificatore a velocità 15 segnala questi frame come marche, mentre quello a velocità 7,5 come ombre. Il classificatore a velocità ridotta riesce a distinguere bene le ombre e, nel caso in cui il frame presenti giunta e ombra, lo classifica correttamente come giunta, dando più importanza a quest'ultima.

Una delle problematiche emerse è lo sbilanciamento del dataset: in particolare ci sono ancora pochi frame relativi alle classi `rovinato` e `segni`, che quindi la rete non riesce ad apprendere correttamente.

In un secondo esperimento [15], si è quindi deciso di non considerare queste ultime due classi per il classificatore a 7.5 ips. Seguendo lo stesso ragionamento, per il classificatore a 15 ips si sono mantenute le sole due classi con un numero elevato di elementi, ovvero giunte e marche.

Dopo aver estratto dal dataset 50 foto per ogni classe, da utilizzare come insieme di test, e dopo aver fatto uno *shuffle*[1] delle immagini, si sono creati il *training set* e il *validation set* secondo la regola 80-20 per ciascuno dei due dataset. I risultati ottenuti da questa rete sul *test set* (composto quindi da 200 foto per il dataset a velocità 7.5 ips e da 100 foto per il dataset a velocità 15 ips) sono plottati nelle matrici di confusione riportate in figura 2.8.

I risultati sono relativi alla classificazione condotta con l'utilizzo di un computer Alienware con CPU 4,00 GHz x8 (Intel Core i7) e una scheda grafica Nvidia GeForce GTX960.

[1] https://scikit-learn.org/stable/modules/generated/sklearn.utils.shuffle.html

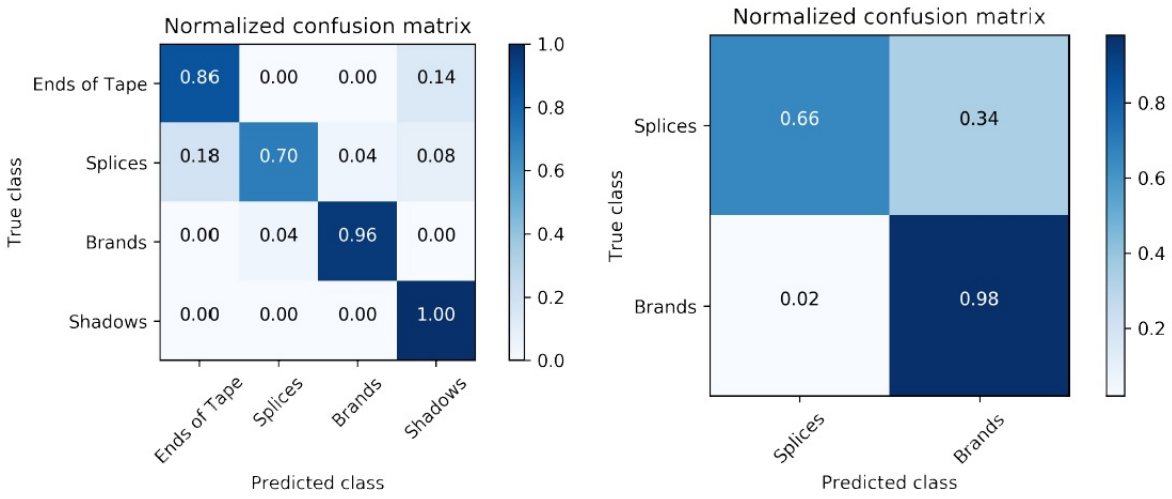


Figura 2.8: *Matrici di confusione: a sinistra per il dataset 7.5, a destra per il dataset 15*

Capitolo 3

Rilevamento discontinuità

Per risolvere le problematiche emerse e descritte in sezione 2.2, si è modificato il software originale, migliorando il codice relativo all'estrazione dei frame. Si è quindi utilizzata per questa prima parte del lavoro di tesi la libreria OpenCV con il suo linguaggio nativo C++.

3.1 Tecnologie utilizzate: OpenCV

OpenCV (*Open source Computer Vision library*), è una libreria software multipiattaforma (Windows, Mac OS, Unix, Android) nell'ambito della visione artificiale. Il linguaggio nativo di tale libreria è C++, ma esistono *wrapper* per Python, Java e Matlab.
OpenCV è rilasciata con licenza BSD (*Berkley Software Distribution*), il cui obiettivo primario è quello di rendere il software completamente libero, accessibile e modificabile da tutti. Chi apporta variazioni a un programma protetto da licenza BSD può ridistribuirlo usando la stessa o un'altra licenza, senza l'obbligo di indicare tutte le modifiche apportate al codice sorgente.

Figura 3.1: *Logo*

Le funzioni messe a disposizione da OpenCV sono circa 500 e sono finalizzate all'elaborazione di immagini, al *tracking* e all'*object detection*, alla calibrazione dei dispositivi, all'estrazione delle *feature* da un'immagine, al riconoscimento di volti, etc. L'immagine può venire rappresentata in bianco e nero, in scala di grigi o a colori: in quest'ultimo caso OpenCV di default utilizza lo spazio di colori BGR, una permutazione dei tre classici canali RGB.

OpenCV è modulare, ovvero è composta da diverse librerie; quelle di interesse per il lavoro di questa tesi sono riportate di seguito.

- *core*: libreria principale di OpenCV; contiene tutte le strutture dati e le funzioni di base per lavorare sulle immagini.

- *imgproc*: libreria con le funzioni per l'elaborazioni di immagini.

- *highgui*: definisce una serie di funzioni per l'I/O e per visualizzare a schermo immagini o video; si occupa in pratica della parte di interfaccia utente.

- *video*: contiene le funzioni per l'analisi dei video.

3.2 Interventi apportati

3.2.1 Selezione ROI

Il primo problema che si è andati a risolvere è stato il metodo di selezione della ROI (*Region Of Interest*), per evitare che l'utente debba trovare e scrivere nel file di configurazione le coordinate numeriche esatte.
Si è optato per creare un'interfaccia con cui l'utente possa interagire e selezionare a piacimento l'area di interesse, lasciando la gestione numerica delle coordinate al software stesso. Prima di far partire l'analisi viene quindi visualizzato a schermo un frame del video, dove l'utente può disegnare un rettangolo per indicare all'applicazione la regione dove eseguire l'analisi dei pixel da confrontare; in caso di errori, l'area di tale rettangolo è modificabile. Una volta selezionata la ROI l'utente può far partire l'analisi, la cui logica è stata mantenuta dallo script originale.

La costruzione del rettangolo è gestita tramite una funzione di *callback* per gli eventi del mouse (*mouse handler*), in ascolto sulla specifica finestra del frame, fino a quando questo non viene chiuso per far partire l'analisi.

Uno dei problemi di questo approccio è la variabilità delle dimensioni del rettangolo, in quanto lasciate decidere all'utente. Più grande è la regione di interesse più bassa dev'essere impostata la soglia per selezionare un frame interessante, altrimenti si rischierebbe di perdere le giunte o i segni più piccoli. Inoltre, è necessario disegnare la ROI in modo che non sia troppo piccola: la distanza percorsa dalla discontinuità tra due frame contigui potrebbe essere infatti maggiore della larghezza del rettangolo.

Dopo una serie di test su nastri e video differenti, si è notato che con una soglia impostata al 97% e l'area della ROI selezionata di altezza pari a quella del nastro e di larghezza calibrata sulla dimensione della testina di lettura (come mostrato in figura 3.2a) si ottengono risultati migliori rispetto al precedente algoritmo (sezione 3.3). Sarà ovviamente necessario istruire l'utente per ottenere i risultati desiderati combinando correttamente il valore della soglia e la dimensione del rettangolo.

In un secondo momento ci si è accorti che alcuni nastri non sono perfettamente allineati rispetto al piano orizzontale; per rendere l'analisi più precisa si è quindi introdotta la possibilità di ruotare la ROI sull'interfaccia: infatti più il rettangolo è calibrato sulla dimensione e sulla posizione del nastro, più l'analisi dei pixel condurrà a risultati precisi (figura 3.2b).
L'utente può attivare l'inclinazione del rettangolo tramite il tasto `Ctrl` e ampliare o diminuire l'angolo di rotazione tramite trascinamento del mouse. Si è quindi modificato il codice della gestione della ROI per adattarla a tutti i possibili casi di rotazione del rettangolo.

(a) **(b)**

Figura 3.2: *Diversi casi per la selezione della ROI*

3.2.2 Riconoscimento fine nastro

Il secondo problema affrontato è stato quello del riconoscimento dell'evento "fine del nastro". Come descritto in sezione 2.2.1, il software rileva tale istante solo se il nastro, una volta finito lo scorrimento, si distacca dalla testina di lettura. In molti casi però tale distacco non avviene; di conseguenza non vi sono modifiche visibili nell'area selezionata come regione di interesse e il frame non viene selezionato come interessante.

Si è cercato quindi un approccio diverso per risolvere il problema, considerando tutti gli elementi effettivamente visibili nel video. La videocamera, infatti, inquadra orizzontalmente il magnetofono, mostrando anche i meccanismi attraverso cui passa il nastro durante la riproduzione (figura 3.3). Di particolare interesse nel nostro caso sono risultati il capstan e il rullo pressore (*pinch wheel*), visibili in figura 3.4.

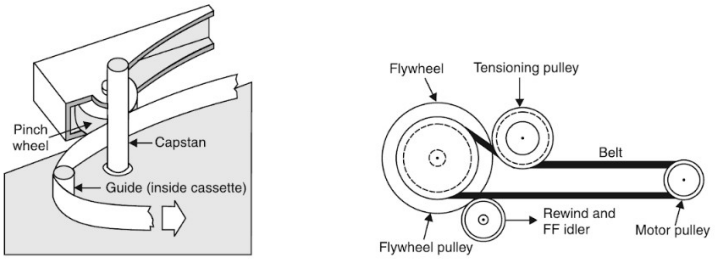


Figura 3.3: *Meccanismi del magnetofono*

Il capstan è un perno rotante utilizzato per spostare il nastro attraverso i meccanismi e le testine magnetiche (di cancellazione, scrittura e lettura) del magnetofono. Durante la riproduzione

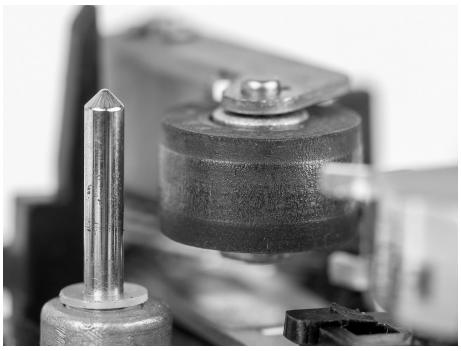


Figura 3.4: *Capstan e rullo pressore*

infatti, il nastro passa attraverso il capstan e una ruota gommata, il rullo pressore: il secondo preme il nastro contro il primo, in modo da fornire l'attrito necessario affinchè il nastro continui a scorrere. Tipicamente il rullo pressore è situato dopo le testine magnetiche nella direzione del nastro in movimento (sulla destra del video nel caso in esame).

L'uso di tali meccanismi permette al nastro di scorrere uniformemente sulle testine, consentendo una velocità precisa e costante nella riproduzione; inoltre, sia il capstan che il rullo pressore devono essere sottoposti a una continua pulizia: qualsiasi irregolarità infatti può causare distorsioni nell'audio, come per esempio *wow* e *flutter*.

Nel momento in cui la riproduzione termina, il rullo rilascia il nastro allontanandosi dal capstan ed è per questo motivo che a volte il nastro esce dalla sede della testina di lettura. Il movimento del rullo pressore è ben visibile in tutti i video e per tale motivo si è scelto di fare affidamento su questo per rilevare l'istante di fine nastro.

È stata quindi introdotta la possibilità di selezionare una seconda regione di interesse, finalizzata a rilevare il suddetto movimento. L'analisi del rettangolo selezionato segue la stessa logica dell'analisi della prima ROI. Per non appesantire computazionalmente e per ridurre quindi i tempi di esecuzione, il controllo del secondo rettangolo selezionato viene però eseguito dall'algoritmo solo nell'ultimo minuto di video, in quanto prima non può capitare l'evento "Fine nastro".

Uno dei problemi sorti con questo approccio è l'eventuale presenza di polveri o impurità sul rullo pressore, se non pulito a dovere: a ogni rotazione queste vengono infatti rilevate come una diversità da notificare e il software salva quindi il frame. Poichè a noi interessa rilevare esclusivamente il movimento, si è notato che non è necessario selezionare il rullo in sè, ma basta selezionare l'area dove il rullo compie il movimento, come mostrato in figura 3.5. Dall'analisi di questo secondo rettangolo viene generato un file di testo contenente le informazioni relative all'evento "Fine nastro" (ovvero il nome del video e il *timestamp*) e il frame viene salvato in un'apposita cartella `finenastro`, in modo tale da essere già classificato.

Figura 3.5: *Selezione ROI per rilevare l'evento "fine nastro"*

3.2.3 Gestione marche

Il problema, già presentato in sezione 2.2, della presenza di marche sul nastro comporta il salvataggio di ogni frame in cui tali stampe sono evidenti, poichè vengono rilevate come qualcosa da notificare (la soglia di pixel diversi tra due fotogrammi consecutivi viene infatti superata). Nel caso in cui le marche siano stampate di continuo su tutto il nastro (figura 3.6) l'algoritmo salva ogni frame che analizza.

Per questo motivo, si è introdotto un controllo ulteriore sui frame da salvare: in fase di configurazione (impostazione della soglia e selezione del video da analizzare) viene chiesto all'utente di indicare se sul nastro sono presenti marche in maniera continua; se presenti, viene attivato un secondo controllo sui frame già ritenuti interessanti, che consiste nel considerare la media del colore presente all'interno del rettangolo in analisi (media aritmetica del colore dei 3 canali RGB per ogni pixel della ROI). In questo modo, se il colore di due frame contigui è simile in media, significa che la differenza rilevata è una marca e l'immagine non viene salvata, altrimenti si tratta di una giunta (il cui colore è diverso da quello del nastro) e si procede regolarmente a salvare il fotogramma. Tale metodo è stato testato con colori che per l'occhio umano sono chiaramente diversi, ma che avrebbero potuto creare problemi all'algoritmo (e.g. giunta bianca su nastro giallo), portando a risultati decisamente migliori rispetto alla precedente versione (come mostrato in sezione 3.3). Inoltre, la prima marca individuata, ovvero la prima immagine che non supera il controllo, viene comunque salvata in un'apposita cartella `marche`.

L'attivazione di tale controllo si è rivelata utile anche nel caso di video con molte variazioni di luce, che portavano al salvataggio di numerosi frame con ombre. Tale metodo non può però essere utilizzato universalmente per tutti i nastri, in quanto andrebbero persi i frame contenenti scritte, segni o frammenti di nastro rovinato, la cui media di colore rimane stabile rispetto a quella dell'intero video.

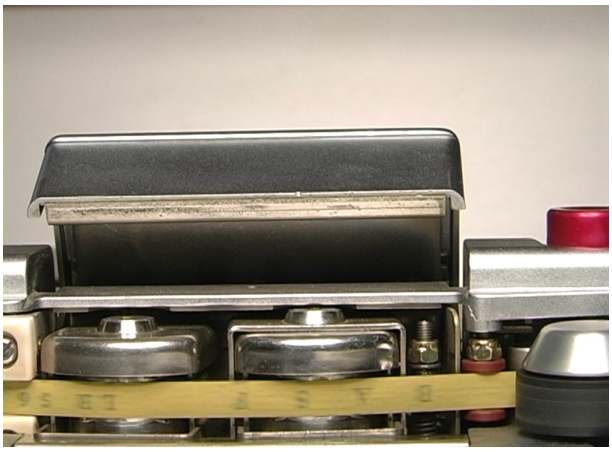


Figura 3.6: *Marca su nastro*

3.2.4 Gestione numero di frame estratti

Come spiegato in sezione 2.2, il software salva numerosi frame per una stessa discontinuità, poichè questa viene rilevata in posizioni diverse del nastro (in ingresso al frame, in posizione centrale e in uscita dal frame).

Tale problema è strettamente collegato alla velocità di riproduzione del nastro. Infatti, più lento è lo scorrimento, più frame della stessa giunta verranno catturati. La tabella 3.1 riporta un esempio del numero di frame salvati per una stessa giunta nello stesso nastro, riprodotto dapprima a una velocità di 7.5 ips e poi di 15 ips: dal primo video l'algoritmo estrae 6 frame, mentre dal secondo 3.

Video	Velocità	Timestamp (ms)	HH:mm:ss
Exp6	7.5 ips	21360	00:00:21
Exp6	7.5 ips	21400	00:00:21
Exp6	7.5 ips	21440	00:00:21
Exp6	7.5 ips	21480	00:00:21
Exp6	7.5 ips	21520	00:00:21
Exp6	7.5 ips	21560	00:00:21
Exp5	15 ips	12680	00:00:12
Exp5	15 ips	12720	00:00:12
Exp5	15 ips	12760	00:00:12

Tabella 3.1: *Frame estratti per una stessa giunta in due video relativi allo stesso nastro riprodotto a velocità diverse*

L'analisi del video avviene a ogni frame e, poichè il video ha un *frame rate* di 25 frame al secondo, vengono estratte immagini anche ogni 40 millisecondi. Questo conduce a un dataset molto rumoroso, con moltissimi frame di scarso interesse, poichè molti si riferiscono alla stessa

discontinuità. Per ridurre il numero di frame estratti, e quindi ridurre il rumore del dataset, si è deciso di lavorare sulla frequenza dei frame analizzati dall'algoritmo.

In un primo momento si è pensato di compiere l'analisi ogni due o più frame, confrontando quindi frame più distanti tra loro, non contigui, ma questo non ha condotto a buoni risultati: i frame salvati erano effettivamente meno, ma qualche giunta non veniva rilevata. Questo perchè se due frame distanti anche 80 (o più) millisecondi presentano entrambi una giunta dello stesso colore il software non si accorge che si tratta di due giunte diverse, il numero di pixel diversi non supera la soglia e il frame non viene salvato.

Si è pertanto passati a un secondo approccio, che si è rivelato più vincente: il calcolo del numero di pixel differenti viene sempre effettuato su frame contigui, ma una volta decretata la presenza di una discontinuità all'interno del frame, quello successivo non viene salvato. Questo ha ridotto parecchio il numero di frame "rumorosi". Si è poi pensato di correlare il numero di frame da non salvare alla velocità del nastro in questione. Pertanto, una volta individuato un frame interessante in un nastro a 7.5 ips verranno saltati i successivi due frame, andando a salvare eventualmente il frame presente 120 ms dopo. Per un nastro a 15 ips, invece, verrà saltato solo il frame successivo, andando a salvare, se necessario, quello presente 80 millisecondi dopo. Per tale motivo, in fase di configurazione viene chiesto all'utente di indicare la velocità di riproduzione del nastro.

3.2.5 Interventi minori

Sono stati infine condotti degli interventi per correggere problemi minori.

- È stato introdotto un messaggio d'errore nel caso in cui il software non trovi il video specificato dall'utente; l'esecuzione quindi si interrompe comunicando la causa dell'insuccesso.
- I frame estratti dall'algoritmo vengono salvati in una cartella chiamata `tmp`; se tale cartella non esiste nel *path* previsto dall'algoritmo, tale cartella viene ora creata. In precedenza l'algoritmo eseguiva l'intera analisi, stampando a terminale il messaggio di salvataggio avvenuto con successo, senza però in realtà salvare nulla.
- Poichè la configurazione da file di testo (in formato XML) è poco pratica per l'utente, è stato scritto predisposto uno script in Python che gestisce l'input dagli utenti dei dati necessari, scrivendoli poi nei file di configurazione; il programma C++ andrà in un secondo momento a leggere tali file.
- In mancanza delle istruzioni per l'installazione dei *tool* necessari per la configurazione o le modalità di utilizzo del *software* è stata scritta la documentazione appropriata (`readme`) per eventuali progetti futuri correlati a tale lavoro.

3.3 Risultati ottenuti

Una volta apportate le modifiche al codice, si è testato il nuovo algoritmo sui video `Esperimento5.mov` (15 ips) ed `Esperimento6.mov` (7.5 ips), che erano stati creati ap-

positamente per giudicare i risultati ottenuti nel lavoro precedente (come già spiegato in sezione 2.2.1) e che contengono un totale di 79 discontinuità.

In tabella 3.2 sono riassunti i risultati ottenuti dall'algoritmo originale e da quello con le modifiche presentate: si può vedere come gli obiettivi inizialmente prefissati siano stati raggiunti.

- Si è aumentata la precisione dell'algoritmo, vengono infatti trovate più discontinuità. In particolare, nel caso dei video in esame:
  - per il video a 7.5 ips il software trova tutte le discontinuità, selezionando due scritte su nastro e l'istante di fine nastro, che l'algoritmo originale non rilevava.
  - per il video a 15 ips vengono rilevate una scritta su nastro in più e la fine del nastro, ma ancora non viene trovata una scritta su nastro.
- È stato ridotto il rumore del dataset: per il video `Esperimento5` vengono salvati quasi la metà dei frame rispetto all'algoritmo originale.
- Sono stati ridotti i tempi di analisi, grazie al fatto che vengono confrontati, e di conseguenza salvati, molti meno frame. Indicativamente, per un video della durata di circa un'ora l'algoritmo conclude l'analisi e l'estrazione dei frame in circa 16 minuti su un PC Alienware con CPU 4,00 GHz x8 (Intel Core i7).
- Si è trovata una soluzione efficiente per il problema delle marche, passando, per un video relativo a un nastro con marche, da un totale di 1368 frame salvati (numero decisamente troppo elevato per lo scopo del progetto) a 4 frame salvati (ovvero le 3 discontinuità presenti più il primo frame relativo alle marche).

Video	Script originale		Nuovo script	
	Frame salvati	Discontinuità trovate	Frame salvati	Discontinuità trovate
Exp5 (15 ips)	205	76	109	78
Exp6 (7.5 ips)	283	76	157	79
Nastro con marche	1368	3	4	3

Tabella 3.2: *Comparazione dei risultati dell'algoritmo di estrazione*

3.4 Creazione nuovo dataset

Una volta testato il funzionamento del nuovo algoritmo, analizzando diversi video e controllando la quantità e la qualità delle discontinuità rilevate, si è deciso di utilizzarlo per ampliare il dataset già esistente (sezione 2.3), con l'obiettivo di di arricchirlo in modo da renderlo effettivamente rappresentativo della realtà in esame, in particolare estraendo nuovi frame per le classi contenenti pochi elementi.

Parallelamente, si è considerata la possibilità di creare un nuovo dataset da utilizzare per le fasi di *training* e *test* della rete neurale. Tale scelta nasce dalla considerazione che l'intero frame, come viene salvato dal precedente algoritmo, contiene moltissimi elementi che non sono di alcun interesse per l'apprendimento e la classificazione che deve compiere la rete neurale: le testine di lettura e scrittura, il capstan, i meccanismi del magnetofono presenti sullo sfondo, etc. Gli unici elementi di interesse sono, di fatto, il nastro stesso e le discontinuità che vi scorrono. Pertanto, si sono considerate le seguenti due opzioni.

- Salvare l'area selezionata dall'utente (ROI).
  Con questo primo approccio, le immagini risultanti sono del tipo in figura 3.7; ovviamente, la discontinuità non viene visualizzata nella sua interezza per due motivi: innanzitutto la ROI potrebbe di fatto essere meno larga dell'intera discontinuità, ma soprattutto l'algoritmo di estrazione salva il frame solo nel momento in cui la quantità di pixel diversi dal frame precedente supera una certa soglia nell'area selezionata. Questo secondo fattore comporta che il frame (e di conseguenza il sotto-frame) venga salvato quando la giunta sta entrando nella ROI, tralasciando i successivi due o tre frame. Utilizzando questo sistema vengono quindi perse le giunte con scritte: per esempio in figura 3.8 è visibile l'area che verrebbe salvata dall'algoritmo e l'informazione persa.
- Salvare l'area relativa all'intero nastro.
  Con questo secondo approccio si è sicuri che l'intera giunta venga salvata, avendo una panoramica dell'intero nastro nel momento in cui la discontinuità viene rilevata (figura 3.9).

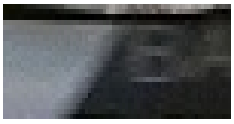


Figura 3.7: *Area ROI*

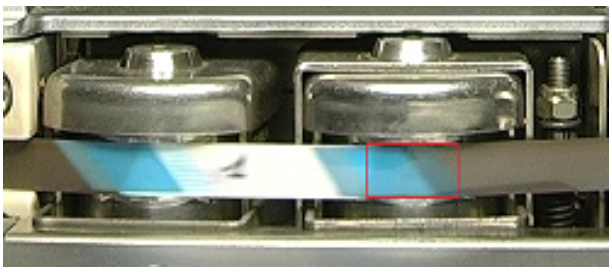


Figura 3.8: *Esempio di giunta con scritta, l'area rossa indica la ROI*

Figura 3.9: *Area relativa al nastro*

L'algoritmo di estrazione è stato quindi ulteriormente modificato per permettere di salvare, oltre all'intero frame, sia i sotto-frame della ROI sia quelli dell'area del nastro, per poter impostare in un secondo momento degli esperimenti e capire quale sia l'approccio migliore da utilizzare per l'allenamento e il test della rete neurale. L'algoritmo estrae l'area del nastro sulla base dell'altezza della ROI selezionata dall'utente e della larghezza dell'intero frame.

In entrambi i casi, prima di venire salvati, i sotto-frame sono stati sottoposti a deinterlacciamento, tramite separazione dei riquadri pari da quelli dispari.

Poiché le dimensioni della ROI sono dettate dalla scelta dell'utente, si sono standardizzate le dimensioni tramite un *resize* delle immagini. In previsione dell'utilizzo della GoogLeNet per la classificazione, si è deciso di effettuare un *resize* a 224x224, dimensione richiesta in input alla rete: l'immagine ottenuta è ovviamente molto distorta, ma le diverse discontinuità sono ancora riconoscibili a occhio umano.

Attualmente, dall'analisi di 70 video sono stati estratti un totale di 4020 frame. Per giungere a un dataset consistente sarà necessario raccogliere ulteriori immagini per poi passare alla successiva fase di allenamento e test di una rete neurale.

Capitolo 4

Classificazione discontinuità

In parallelo alla creazione del nuovo dataset e in previsione dell'implementazione di una nuova rete da allenare, si è deciso di testare le nuove tecnologie scelte implementando nuovamente la GoogLeNet utilizzando il dataset esistente, con l'obiettivo di riprodurre l'esperimento presentato in sezione 2.3. Al posto di Caffe, infatti, come framework per l'implementazione è stato scelto Keras, usando come *back-end* TensorFlow.

Nella prossima sezione verrà fatta una breve introduzione alle reti neurali convoluzionali per poi passare alla descrizione delle tecnologie utilizzate.

4.1 Reti neurali convoluzionali

Allo stato dell'arte, per la classificazione di immagini vengono utilizzate soprattutto le reti neurali convoluzionali, comunemente chiamate CNN o ConvNet, uno speciale tipo di ANN (Artificial Neural Network) che elaborano dati con una topologia a griglia nota, per esempio:

- griglia su spazio (immagini 2D)
- griglia nel tempo (segnali 1D)

Una rete neurale per la classificazione di immagini riceve e gestisce matrici di pixel come input; poichè i valori dei pixel sono RGB, si vanno a gestire tre matrici per ogni immagine, una per ogni componente di colore.

Le reti neurali convoluzionali seguono la struttura classica di una generica rete: sono composte da un *layer* di input, uno o più *layer* intermedi (detti nascosti) e un *layer* di output. Ogni *layer* è costituito da neuroni, i quali possiedono un insieme di ingressi, un'uscita e una funzione di attivazione (neurone "acceso" o "spento"). I neuroni sono collegati tra loro tramite connessioni pesate, alle quali viene assegnato un valore w per ogni input, fornendo una misura di quanto è importante tale input nel neurone (figura 4.1).

In fase di allenamento la rete apprende, aggiornando ad ogni iterazione tali pesi. Il metodo più utilizzato è quello della discesa del gradiente (*gradient descend*), una tecnica di retropropagazione.

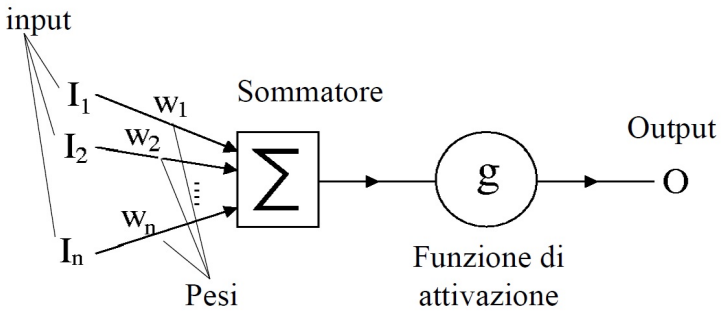


Figura 4.1: *Modello matematico del neurone*

Dopo aver propagato l'input attraverso i diversi strati, la rete produce un output, in base al quale compie delle predizioni sui dati (fase di *forward propagation*). A questo punto, viene calcolata una *loss function* $L(X_i, w)$, che quantifica quanto l'output prodotto all'iterazione i-esima si discosta da quello desiderato. Tale errore può essere ridotto modificando i pesi w in direzione opposta al gradiente di L (regola di derivazione a catena). Infatti, il gradiente indica la direzione di maggior crescita di una funzione in più variabili e muovendosi in direzione opposta si riduce l'errore.

Come si evince dal nome, però, le CNN adottano la convoluzione, concetto matematico utilizzato nel *Digital Signal Processing* che consiste nel combinare due funzioni (variabili nel tempo) in maniera coerente. L'operazione nel dominio discreto per due variabili viene definita come segue:

$$(f * g)(x, y) \sum_{u=-\infty}^{\infty} f(u, v) \cdot g(x - u, y - v) \tag{4.1}$$

Quella che effettua la rete, in realtà, è una convoluzione 3D: il filtro infatti opera su una porzione di volume e non di superficie.

Di seguito, vengono analizzati i diversi *layer* e le funzioni di attivazione che compongono l'architettura di una CNN generica.

Layer convoluzionale

Il processo di convoluzione è ispirato dai processi biologici per l'analisi visiva negli organismi viventi. Lo strato di neuroni che si occupa della convoluzione divide l'immagine in vari frammenti sovrapposti, che sono in seguito analizzati per individuare dei *pattern* caratterizzanti, trasferendo l'informazione allo strato seguente sotto forma di una *feature map*. In sostanza, l'algoritmo cerca di rilevare all'interno dell'immagine delle *feature*, ovvero caratteristiche che contraddistinguono i diversi elementi: se, ad esempio, una rete ricevesse come input diverse foto di volti di persone, cercherebbe di trovare delle caratteristiche comuni, come due occhi, due orecchie, un naso, una bocca. Uno dei componenti principali della rete sono i filtri (o *convolution kernel*): dal punto di vista pratico corrispondono a piccole matrici (in relazione alle dimensioni delle immagini) contenenti dei valori. Viene eseguita la convoluzione di tale filtro con l'input,

ottenendo come risultato le *activation maps*, ovvero le regioni nelle quali sono state individuate delle *feature*. I valori presenti all'interno di ognuna di queste matrici cambia ad ogni iterazione sul *training set*, poichè la rete sta imparando a riconoscere e ad estrarre una certa *feature* dall'immagine.

Il livello di convoluzione però introduce:

- complessità, poichè analizza piccole parti dell'input, di cui molte sovrapposte tra loro (*overlapping*) e di conseguenza il volume in output è molto più grande dell'input e di livello in livello la dimensionalità diventa intrattabile;
- *overfitting*, poichè il modello potrebbe imparare caratteristiche che sono specifiche solo del *training set*, ma che non hanno riscontro nel resto dei casi.

A questi problemi si fa fronte introducendo uno strato di *pooling* consecutivamente allo strato convoluzionale.

Pooling Layer

Il *pooling* è un processo di sottocampionamento (detto anche decimazione) alquanto comune nelle reti neurali: esso consiste nel ridurre la dimensione dei dati, in modo da rendere più rapida l'analisi. Nel caso di un'immagine, il processo è molto simile a una riduzione di qualità dei pixel: una regione di pixel, detta finestra, viene ricondotta a un unico pixel, al quale è assegnato un colore in base alla media dei colori della regione di partenza (*average pooling*). In alternativa, si può scegliere di assegnare al nuovo pixel il valore massimo da ogni regione (*max pooling*) (figura 4.2). Solitamente, il *pooling layer* è posto consecutivamente a un *layer* convoluzionale, in modo da ridurre la dimensione dell'input da passare allo strato successivo.

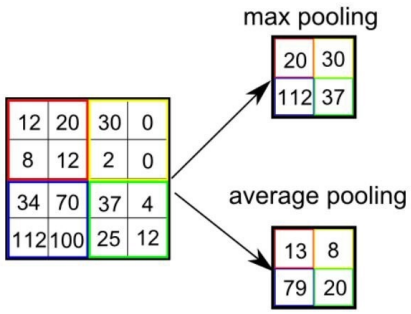


Figura 4.2: *Esempio di max e averaging pooling*

Funzioni di attivazione

Le funzioni di attivazione sono necessarie per trasformare l'ampiezza dell'output di un neurone, che sarà dato in ingresso a un neurone dello strato successivo. Nelle reti MLP (Multi-Layer Perceptron) la funzione di attivazione più utilizzata è la sigmoide:

$$f(v_k) = \frac{1}{1 + e^{v_k}} \tag{4.2}$$

Dove v_k è l'input al neurone k-esimo.

Nelle reti profonde l'utilizzo di tale funzione è però problematico, a causa della retropropagazione del gradiente. La rete neurale infatti, come già detto, aggiorna i pesi proporzionalmente alla derivata parziale (gradiente) della funzione rispetto al peso corrente in ciascuna iterazione di allenamento. La derivata della sigmoide è tipicamente minore di 1, quindi l'applicazione della regola di derivazione a catena porta a moltiplicare tra loro molti termini minori di 1, riducendo di conseguenza i valori del gradiente nei livelli lontani dall'output. La sigmoide ha inoltre un comportamento saturante verso gli estremi (0 o 1), col rischio che il suo gradiente di conseguenza si annulli (problema della scomparsa del gradiente).

Per attenuare tale problema, si utilizza la funzione ReLu (Rectified Linear Unit):

$$f(v_k) = max(0, v_k) \tag{4.3}$$

Tale funzione satura solo nel caso in cui l'input sia minore di 0; inoltre è più efficiente computazionalmente, poichè si tratta di scegliere il massimo tra due valori e non di eseguire operazioni particolarmente pesanti, come per esempio l'esponenziale della sigmoide.

Fully connected Layer

L'ultimo livello della rete è una classica rete MLP completamente connessa. Il livello finale utilizza una funzione di attivazione diversa chiamata *softmax* (funzione normalizzata esponenziale); per ogni neurone k esegue una sorta di normalizzazione, i cui valori di uscita z_k possono essere interpretati come una probabilità (sono tra 0 e 1 e la loro somma è 1):

$$z_k = f(v_k) = \frac{e^{v_k}}{\sum\limits_{i=1,\dots,s} e^{v_i}} \tag{4.4}$$

L'ultimo layer si occupa di calcolare la *loss function*, che misura quanto sono sbagliate le predizioni della rete rispetto al vero *label*. Al posto di utilizzare la somma degli errori quadratici medi per misurare l'accuratezza (come nelle MLP), si utilizza come funzione di costo la *cross-entropy* tra due distribuzioni discrete p e q, che fissata p misura quanto q differisce da p:

$$H(p, q) = \sum_{v} p(v) \log q(v) \tag{4.5}$$

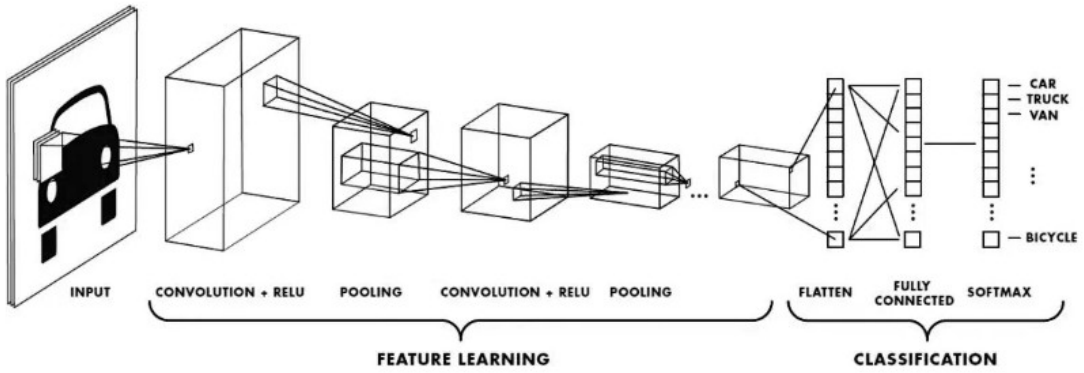


Figura 4.3: *Architettura CNN*

4.2 Tecnologie utilizzate

Al momento di scegliere che tecnologie utilizzare per la fase di classificazione di immagini, sono state considerate due possibili opzioni: configurare la macchina e lavorare direttamente su di essa oppure utilizzare la *containerizzazione*, tramite Docker. Il lavoro è stato svolto seguendo la prima modalità, ma è stata configurata la macchina anche per utilizzare Docker. Nei seguenti paragrafi verranno presentate tutte queste tecnologie.

4.2.1 Keras

Keras è una libreria *open source* ad alto livello per l'apprendimento automatico e le reti neurali, scritta in Python. È progettata come un'interfaccia a un livello di astrazione superiore rispetto ad altre librerie simili e supporta come *back-end* le librerie TensorFlow, Microsoft Cognitive Toolkit (CNTK) e Theano. Progettata per permettere una rapida prototipazione di reti neurali profonde, è stata sviluppata come parte del progetto di ricerca ONEIROS e il suo autore principale è François Chollet di Google.

Figura 4.4: *Logo*

I tre vantaggi principali di Keras sono:

- **Facilità d'uso**, Keras ha un'interfaccia semplice, ottimizzata per casi di uso comune, che fornisce un *feedback* chiaro e attuabile per gli errori degli utenti;
- **Modularità**, i modelli Keras sono realizzati collegando insieme blocchi predefiniti configurabili, con poche restrizioni;
- **Estensibilità**, permette di scrivere elementi costitutivi personalizzati, per esprimere nuove idee per la ricerca; permette inoltre di creare reti personalizzate, creando nuovi *layer* e *loss function*, nonché di sviluppare modelli dello stato dell'arte.

4.2.2 TensorFlow

Figura 4.5: *Logo*

TensorFlow è una libreria *software open source* per l'apprendimento automatico che fornisce moduli testati e ottimizzati utili alla realizzazione di algoritmi per diversi tipi di compiti percettivi e di comprensione del linguaggio. È una seconda generazione di API, attualmente usata sia in ambito di ricerca sia in ambito di produzione in diversi prodotti commerciali Google come il riconoscimento vocale, Gmail o Google Foto. TensorFlow fu sviluppato dal team Google Brain e rilasciato sotto la licenza *open source* Apache 2.0 nel 2015.

Come indicato dal nome, il framework definisce ed esegue computazioni coinvolgendo i tensori, ovvero una generalizzazione di vettori e matrici a dimensioni molto elevate: essi sono di fatto rappresentati come array n-dimensionali. Ogni tensore è caratterizzato da un rango (*rank*), cioè il numero di dimensioni nello spazio, e da una forma (*shape*), cioè il numero di elementi per ogni dimensione.
Per esempio: $[[[1., 2., 3.]], [[7., 8., 9.]]]$ è un tensore di rango 3 con forma $(2, 1, 3)$.

Le immagini che la rete andrà a classificare sono rappresentate da tensori di rango 3 e forma $(224, 224, 3)$, definita dalle dimensioni in pixel e dai tre canali RGB.

Rispetto a Caffe, TensorFlow guadagna sempre più popolarità tra gli sviluppatori per il *deep learning*, in quanto è più facile da implementare e gode di un'API più flessibile.

Altri aspetti che favoriscono il framework di Google rispetto a Caffe sono i seguenti.

- Distribuzione: Caffe non ha un modello semplice per l'installazione, al contrario di TensorFlow che ha il gestore di pacchetti `pip` (acronimo ricorsivo, che significa *Pip Installs Packages*) per Python.

- API: Caffe manca di un'API di alto livello per la costruzione dei modelli, rendendo più complessa la configurazione e la sperimentazione. Inoltre, TensorFlow ha un'interfaccia più adatta per Python, linguaggio sempre più scelto tra gli sviluppatori, poichè riduce l'attività manuale rispetto a C++, linguaggio nativo di Caffe.

- GPU: poiché il supporto per GPU di Caffe attualmente non offre strumenti per Python, tutto l'allenamento deve essere eseguito tramite un'interfaccia a linea di comando basata su C++. TensorFlow invece permette una gestione molto più immediata, tramite `tf.device()`, in cui si indica l'uso delle GPU. Non è necessaria ulteriore documentazione, né ulteriori modifiche alle API. TensorFlow offre inoltre un'architettura più flessibile in quanto è possibile eseguire due copie di un modello su due GPU o un singolo modello di grandi dimensioni su due GPU.

Nel 2017 è stato introdotto il supporto per Keras: il modulo `tf.keras` è l'implementazione delle API Keras come libreria di alto livello per TensorFlow.

4.2.3 Docker

Docker[1] è uno strumento *open source* designato con l'obiettivo di rendere più semplice la creazione, lo sviluppo, l'esecuzione e la distribuzione di applicazioni utilizzando la tecnologia dei *container*. Questi permettono allo sviluppatore di isolare un'applicazione con tutte le dipendenze da essa richieste (per esempio le librerie necessarie), indipendentemente dal sistema operativo utilizzato. Docker infatti è nativo per Linux ma è disponibile anche per piattaforma OS e Windows.

Figura 4.6: *Logo*

La tecnologia Docker utilizza il kernel di Linux e le sue funzionalità per isolare i processi in modo da poterli eseguire in maniera indipendente. Questa indipendenza è l'obiettivo dei *container*: la capacità di eseguire più processi e applicazioni in modo separato per sfruttare al meglio l'infrastruttura esistente, senza il vincolo di ogni altra personalizzazione possibile delle impostazioni della macchina sulla quale verrà eseguito il programma.

Gli strumenti per la creazione di *container*, come Docker, consentono il *deployment* a partire da un'immagine, ovvero un *template* di sola lettura con le istruzioni necessarie. Si può utilizzare un'immagine creata da altri e pubblicata in *repository* ufficiali, oppure si può definire la propria immagine, creando un file chiamato Dockerfile, dove ogni istruzione crea un livello nell'immagine. Quando questo file viene modificato solo i livelli che sono stati cambiati vengono ricostruiti, evitando quindi uno spreco inutile di risorse. Questo rende le immagini più leggere e veloci rispetto ad altre tecnologie di virtualizzazione.

I *container* invece sono le istanze avviabili di un'immagine. Si possono creare, avviare, fermare, spostare o cancellare utilizzando le API di Docker. Si possono inoltre connettere a una o più reti o assegnare loro uno spazio di *storage*.

Lo svantaggio di questi sottosistemi leggeri risiede nella vulnerabilità creata nel momento in cui il kernel dell'host viene condiviso con i *container*. Ciò non si verifica nel caso delle macchine virtuali, che sono molto più isolate dal sistema host. L'isolamento quindi non è totale e le risorse a disposizione del *container* sono meno garantite rispetto a quelle fornite senza livello di virtualizzazione.

Per l'allenamento della rete con *performance* migliori è necessario servirsi della GPU; è possibile utilizzare quest'ultima dentro al *container* tramite Nvidia-Docker (al momento alla versione 2)[2], che gestisce il collegamento tra la scheda grafica Nvidia e Docker utilizzando CUDA[3] (Compute Unified Device Architecture), come mostrato in figura 4.7. Esistono delle immagini Docker ufficiali da scaricare, già configurate per l'utilizzo di TensorFlow con il supporto per la GPU, tra cui `latest-gpu-py3`, la quale ha il vantaggio di non dover installare CUDA in locale (piuttosto oneroso e complicato da gestire), poichè la configurazione è già inclusa nell'immagine stessa. I comandi per scaricare l'immagine ed eseguire il *container* sono i seguenti:

[1] https://docs.docker.com/install/linux/docker-ce/ubuntu/
[2] https://github.com/NVIDIA/nvidia-docker
[3] https://developer.nvidia.com/cuda-zone

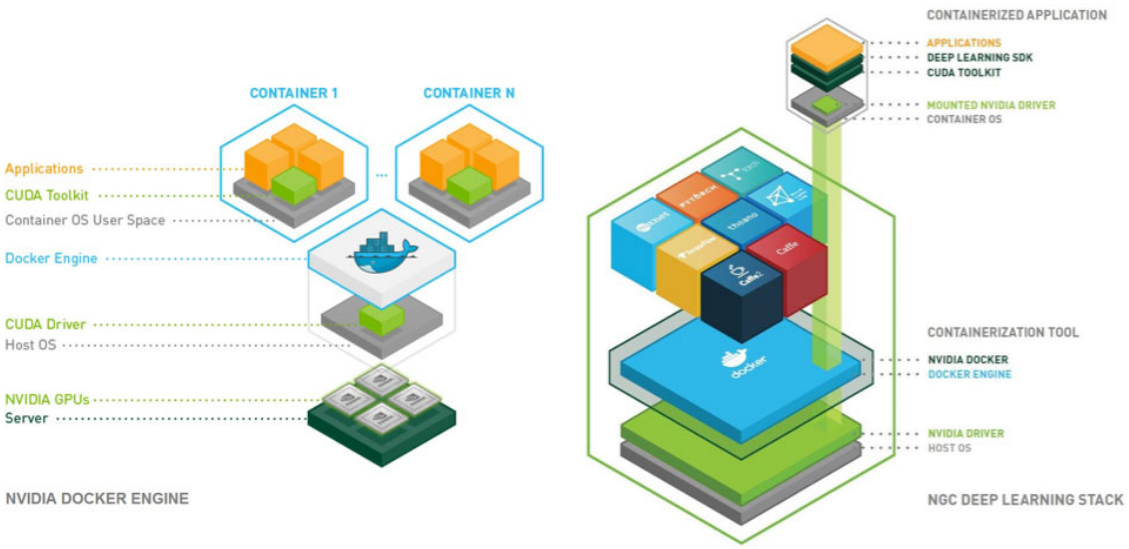


Figura 4.7: *Nvidia-Docker*

```
$ docker pull tensorflow/tensorflow:latest-gpu-py3
```

```
$ docker run --rm --runtime=nvidia -it -p 8888:8888 tensorflow/
    tensorflow:latest-gpu-py3
```

Quest'ultimo comando automaticamente lancia il server `jupyter notebook`[4] su `localhost:8888`.

Per salvare i *notebook* direttamente in locale è sufficiente fare il *mount* della cartella di lavoro, aggiungendo il parametro `volume`:

```
--volume="/home/$USER:/notebooks/$USER"
```

4.2.4 GooogLeNet

La rete GoogLeNet [16] è stata la vincitrice della competizione ILSVRC 2014 (ImageNet Large Scale Visual Recognition Challenge), risultando la più efficiente tra quelle presentate, ottenendo come accuracy top-1 68.7% e accuracy top-5 88.9%. La rete è costituita da un centinaio di *layer* ed è stata allenata e testata su un dataset suddiviso in 1.2 milioni di immagini per il *training set*, 50.000 per il *validation set* e 100.000 per il *test set*. La rete è stata allenata con immagini di dimensione $(224, 224, 3)$ per un totale di 1000 classi in output.

[4] https://jupyter.org/

Architettura

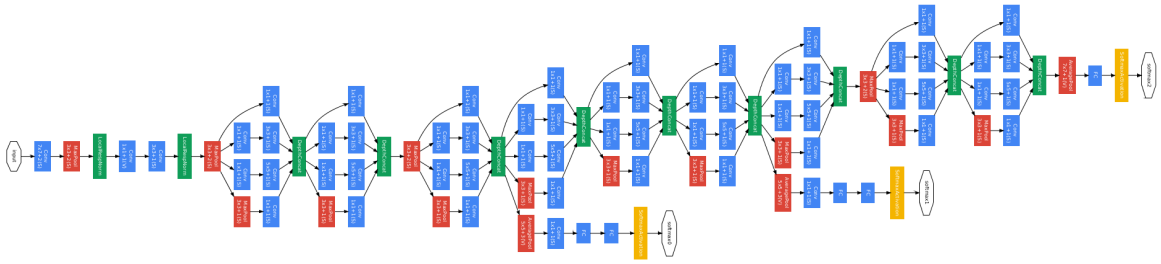
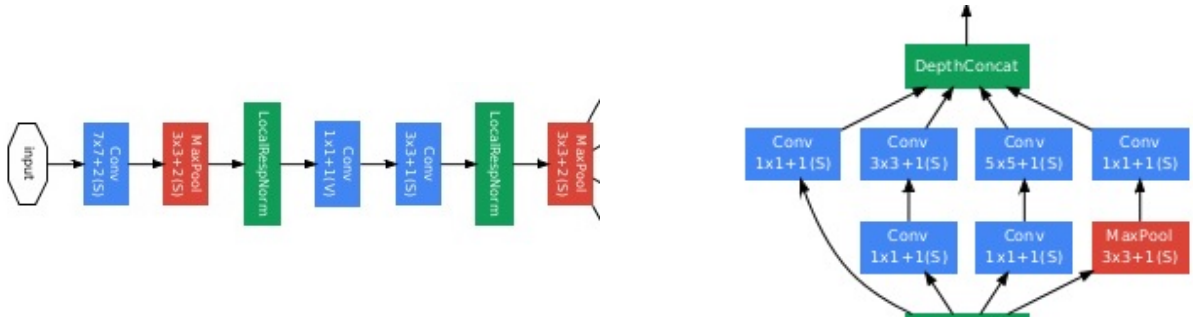


Figura 4.8: *Architettura di GoogLeNet*

I primi strati, attraverso cui deve passare l'immagine in input alla GoogLeNet, sono una serie di layer convoluzionali, di *pooling* e di normalizzazione, come le classiche reti neurali convoluzionali. Questo primo blocco viene definito `stem` (figura 4.9a).

A questo segue il blocco di base della rete di Google, il modulo *inception* (figura 4.9b), composto da una serie di *layer* convoluzionali e di *pooling*, posti in parallelo e poi concatenati tra loro. La rete contiene 9 di questi moduli inframmezzati da due *layer max pooling*, per ridurre le dimensioni delle matrici che attraversano la rete.



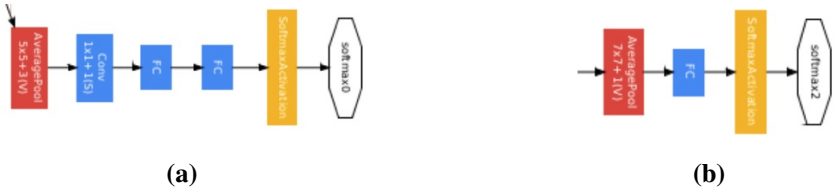
(a) **(b)**

Figura 4.9: *I moduli Stem (a) e Inception (b)*

All'interno del modulo *inception* vengono utilizzate le convoluzioni 1×1, per ridurre la dimensionalità delle matrici in input alle convoluzioni poste successivamente, le quali presentano dimensioni di filtro maggiori (5×5 o 3×3). Questo approccio si traduce in un modello ad alte prestazioni con un numero minore di parametri rispetto ad altre reti convoluzionali.

A causa della profondità della rete, sono stati inseriti due output ausiliari (figura 4.10a), uno a circa metà e l'altro a due terzi della rete. Lo scopo di questi classificatori è di attenuare il problema della scomparsa del gradiente, amplificando il segnale del gradiente attraverso la rete.

Infine, si trova l'output classificatore (figura 4.10b), formato da un *layer average pooling* seguito da una funzione di attivazione *softmax* su un *fully connected layer*.



(a) **(b)**

Figura 4.10: *I tre output della GoogLeNet: due ausiliari (a) e uno normale (b)*

In Appendice B è riportata l'implementazione della GoogLeNet in Keras-TensorFlow. Tutti e tre gli output della rete sono stati modificati per classificare 4 classi per il dataset 7.5 e 2 classi per il dataset 15.

4.3 Allenamento rete

4.3.1 Training, Validation e Test set

Per riprodurre il più fedelmente possibile l'esperimento condotto con la vecchia rete neurale, presentato in [15] e in sezione 2.3, si sono utilizzati gli stessi dati. In particolare, per ciascuno dei due dataset sono stati rimescolati i dati (tramite la funzione *shuffle*) e sono state estratte in maniera random 50 foto da ogni classe: `fine nastro`, `giunte`, `marche` e `ombre` per il dataset 7.5 e solamente `giunte` e `marche` per il dataset 15.

Le rimanenti immagini sono state divise in *training set* e *validation set* secondo la regola 80-20, tramite la funzione della libreria Sklearn $train\_test\_split$, ottenendo quanto riportato in tabella 4.1.

Per rendere riproducibile l'esperimento è sufficiente inserire, per entrambe le operazioni sopra descritte, il parametro $random\_state$, che permette di ottenere sempre la stessa sequenza randomizzata. Per tale motivo, le 50 foto estratte da ogni classe e la divisione del dataset rispecchiano perfettamente l'esperimento precedentemente condotto.

Dataset	Training set	Validation set	Test set	Totale elementi
7.5	8611	2013	200	10.964
15	6226	1557	100	7.721

Tabella 4.1: *Divisione degli elementi del dataset in training, validation e test set*

4.3.2 Preprocessing

Una volta creati i tre set su cui lavorare, si è eseguito il *preprocessing* delle immagini da dare in ingresso alla rete neurale. È necessario prestare attenzione a eseguire lo stesso procedimento per tutti e tre i set, altrimenti in fase di test i dati non sono consistenti.

Il primo step, in fase di lettura dei dati dal disco, è quello di effettuare un *resize* delle immagini alla dimensione richiesta in input dalla rete, ovvero $(224, 224, 3)$.

Si è passati poi alla normalizzazione dei dati; a volte tale operazione può infatti aumentare le performance delle reti. Essa consiste nello scalare i dati nella stessa maniera e centrarli rispetto una determinata media.
Pertanto, come riportato nell'estratto di codice 4.1, ogni immagine è stata convertita in rappresentazione *floating point* a 32 bit, per poi dividere ogni valore del tensore per 255, in modo tale da ottenere matrici di valori compresi tra 0 e 1.

```
for img in X_train:
        img.astype('float32')
X_train = np.stack(X_train, axis=0)
X_train = X_train/255
```

Codice 4.1: *Normalizzazione dei dati*

4.3.3 Training

Per allenare la rete vanno configurati alcuni parametri, necessari alla rete per capire come gestire i dati in input.

Terminologia e parametri utilizzati

- **Iterazione**: un *forward pass* e un *backward pass* di un campione di dati formano un'iterazione. Nella prima fase l'input attraversa tutti i *layer* e viene calcolata la *loss function*; nella seconda fase, invece, la rete aggiusta i pesi e, di fatto, impara.

- **Epoca**: descrive il numero di volte in cui l'algoritmo osserva l'intero training set; consiste in un *forward pass* e un *backward pass* di tutti i dati di training.

- **Batch size**: il numero di campioni di training che l'algoritmo osserva in un'iterazione. Più alto è posto tale valore, più spazio in memoria è necessario.

  Per esempio, se si hanno 1000 campioni di training e si imposta un *batch size* pari a 500, saranno necessarie due iterazioni per completare un'epoca.

- **Learning rate**: l'obiettivo della rete neurale è quello di minimizzare la *loss function*; matematicamente, se la funzione di costo è $L(X_i, w)$, dove X_i indica l'i-esima iterazione e w i pesi che la rete apprende, la rete aggiorna i pesi secondo: $w := w - \eta \nabla L$

η è chiamato *learning rate* (tasso di apprendimento) e determina quanto velocemente la rete aggiorna i parametri. Solitamente si imposta un learning rate iniziale (circa 0.001) e lo si aggiorna ciclicamente dopo un certo numero di iterazioni, moltiplicandolo per una costante minore di 1.

- **Ottimizzatore**: gli ottimizzatori aggiornano il modello cercando di adattarlo a seconda dell'output della funzione di costo. Come spiegato in 4.1, uno degli ottimizzatori più usati è la discesa del gradiente (*Gradient Descend* - GD).

  A volte può essere più efficiente utilizzare, a ogni iterazione, solo un sottoinsieme dei campioni di training per calcolare il gradiente. La discesa del gradiente stocastico (*Stochastic Gradient Descend* - SGD) è un'implementazione che utilizza *batch* di campioni a ogni iterazione, calcolando quindi una stima del gradiente.

- **Regolarizzazione**: la regolarizzazione è un termine aggiunto nel processo di ottimizzazione che aiuta a evitare l'*overfitting*, aggiungendo alla funzione di costo una penalizzazione per pesi elevati in modo da generalizzare meglio i nuovi dati. Una delle tecniche di regolarizzazione è il *dropout*, che consiste nell'impostare in maniera random una frazione di unità di input a 0 ad ogni aggiornamento durante l'allenamento.

Il training sui dati è stato quindi eseguito con le impostazioni riportate in tabella 4.2; il valore del *batch size* è stato impostato seguendo la capacità della GPU.

Numero epoche	50
Batch size	30
Learning rate iniziale	0.001
Costante moltiplicativa	0.96
Cadenza diminuzione LR	8 epoche
Ottimizzatore	SGD
Tecnica di regolarizzazione	Dropout

Tabella 4.2: *Configurazione training*

Di seguito sono riportati i comandi per creare e allenare il modello con Keras, indicando tutti i parametri appena presentati.

```
model.compile(loss=['categorical_crossentropy','
    categorical_crossentropy','categorical_crossentropy'],
    loss_weights=[1,0.3,0.3],optimizer=sgd,metrics=['accuracy']

model.fit(X_train,[y_train,y_train,y_train],validation_data=(
    X_val,[y_val,y_val,y_val]),shuffle=True,epochs=50,batch_size
    =30,callbacks=[lr_sc])
```

Codice 4.2: *Creazione modello e Training*

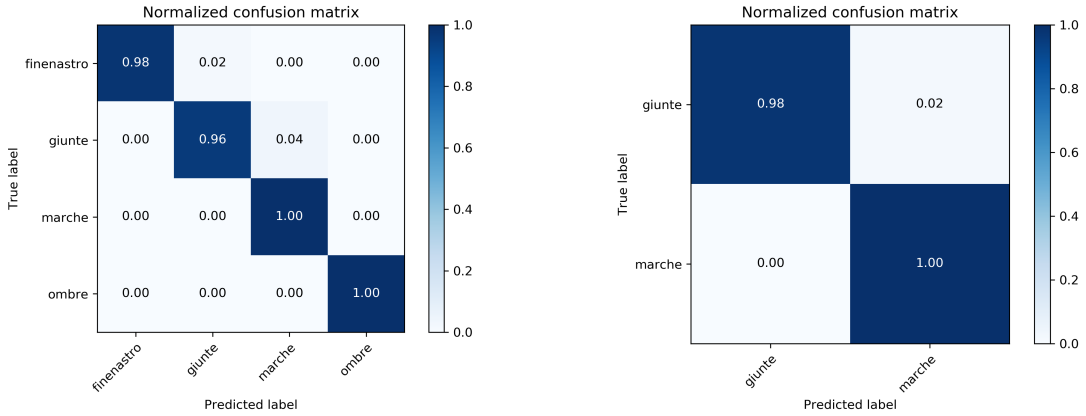
Indicativamente, con la quantità di dati a nostra disposizione, il training di un'epoca su un PC Alienware con CPU 4,00 GHz x8 (Intel Core i7) e una scheda grafica Nvidia GeForce GTX960 avviene in circa 100 secondi.

Alla fine di questa fase l'architettura del modello viene salvata in un file JSON, mentre i pesi che la rete ha assegnato vengono salvati in un file `model.h5`; entrambi tali file vengono caricati in fase di test, prima di eseguire la predizione sui dati.

4.3.4 Risultati ottenuti

Riproduzione esperimento

Entrambi i classificatori in fase di *validation* ottengono accuratezze del 99%, dato confermato anche dalla successiva fase di test sulle foto estratte dal dataset prima di allenare la rete. In figura 4.11 sono riportate le matrici di confusione, plottate in seguito al test sulle stesse immagini su cui era stato eseguito l'esperimento descritto in [15] (sezione 2.3): vediamo come la nuova rete implementata conduca a risultati migliori rispetto alla precedente.



(a) **(b)**

Figura 4.11: *Accuratezze ottenute con la nuova rete sui test set provenienti dal dataset 7.5 (a) e dal dataset 15 (b)*

Esperimento con nuovo test set

Avendo ottenuto delle accuratezze molto buone sulle immagini presenti nel dataset, si è passati a un test successivo: utilizzare i classificatori per decretare la tipologia di immagini estratte da video che la rete non ha mai visto in precedenza.

Tramite il metodo di Keras `predict` la rete calcola, per ciascuno dei suoi tre output, la predizione; considerando solo l'output finale della rete, tralasciando quelli ausiliari, la predizione risulta una matrice le cui righe (corrispondenti alle immagini) sono vettori di valori compresi tra 0

e 1, indicanti la probabilità di appartenenza dell'immagine a ciascuna classe. Il risultato si estrae prendendo il valore di probabilità maggiore, come mostrato in Codice 4.3, ottenendo quindi un vettore contenente tutte le predizioni per ogni immagine.

```
pred = model.predict(X_test, batch_size=30)
prediction = np.argmax(pred[0], axis=1)
```

Codice 4.3: *Prediction su dati nuovi*

Tale esperimento, i cui risultati per il dataset 7.5 sono riportati nella matrice di confusione in figura 4.12a, evidenzia come il dataset non sia adeguato a generalizzare il problema, confermando la necessità di procedere alla creazione di un nuovo dataset, contenente più elementi e formato da immagini del nastro, piuttosto che dell'intero frame. La rete tende infatti a classificare sia le marche che le ombre come giunte, riuscendo invece a identificare abbastanza bene i fine nastro, con un'accuratezza del 75%. Da un'analisi accurata del dataset 7.5 è emerso che la classe `ombre` è stata creata utilizzando frame estratti dagli stessi tre video e che la classe `marche` è composta da 5840 frame, di cui 4438 provengono dallo stesso video. Questo sicuramente comporta il fatto che la rete apprenda molto bene i casi specifici, ottenendo un'accuratezza molto alta su un test set estratto dallo stesso dataset sui cui la rete è stata allentata, ma un'accuratezza bassa su un test set costruito con un caso generico; per esempio, se il nastro del video da cui sono stati estratti i 4438 frame è rosa, la rete tenderà ad abbinare i nastri rosa alla classe `marche`, nonostante non via sia alcuna correlazione tra i due elementi.

Lo stesso ragionamento vale per il dataset 15. La classe `marche`, contenente 5709 frame, è stata creata con pochi video: in particolare, 3166 sono provenienti da un video e altri 1708 da un secondo video, ottenendo immagini decisamente troppo simili. La rete, quindi, non riesce a generalizzare il concetto di marca. I risultati su un test set formato da nuovi elementi portano infatti la rete a classificare al 100% tutte le immagini come giunte, i cui campioni di *training* sono invece di provenienza di video diversi.

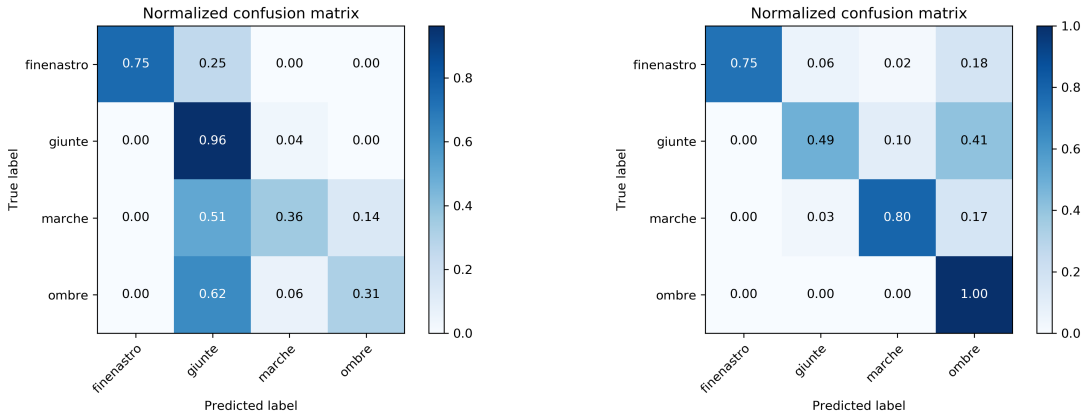
Nuovi esperimenti sul dataset 7.5

Si sono quindi tentati diversi approcci per giudicare i comportamenti di diversi classificatori sui video sconosciuti col classificatore 7.5: in un primo momento si è creato un nuovo classificatore per riconoscere solamente tra giunte e ombre; in questo caso la rete tende a classificare le diverse immagini come ombre, con un'accuratezza dell'80% sulle ombre e del 30% sulle giunte.

In seguito, si è allenata la rete sullo stesso dataset pre-processando ulteriormente le immagini, trasformandole in scala di grigio; in questo caso la rete ottiene risultati leggermente migliori su tre classi, non riconoscendo però le marche.

Si è cercato poi di intervenire sul dataset, cercando di riequilibrare le diverse classi aggiungendo frame da altri video, ottenendo però ancora una volta che la rete tenda a classificare i diversi frame come ombre.

Trasformando successivamente le immagini di questo secondo dataset in scala di grigi si sono ottenuti i risultati in figura 4.12b. La rete tende ancora a classificare le giunte come ombre nel 40% dei casi ma notiamo un miglioramento sulle rimanenti classi.



(a) **(b)**

Figura 4.12: *Risultati su frame estratti da video sconosciuti alla rete, prima (a) e dopo (b) gli interventi effettuati sul dataset 7.5*

Esperimento con il nuovo dataset

Dall'analisi di circa 70 video a diverse velocità è stato creato un unico dataset formato dalle immagini dei sotto-frame relativi all'area del nastro (come spiegato in sezione 3.4). È stato scelto di non dividere a seconda della velocità, in quanto la differenza tra i frame dei due dataset non è così incisiva.

Le classi scelte per il nuovo dataset sono `giunte`, `ombre` e `marche`. La classe `finenastro` è stata rimossa in quanto, dopo le modifiche apportate all'algoritmo di estrazione, il frame per questo evento viene salvato immediatamente in un'apposita cartella. È stato scelto di includere comunque la classe `marche` nella classificazione, in quanto il controllo sul colore presentato in sezione 3.2 può non venire attivato: le marche possono infatti comparire solo in alcuni tratti del nastro, col conseguente salvataggio dei frame da parte dell'algoritmo.

Classificando quindi manualmente le immagini, si è ottenuto un nuovo dataset, suddiviso come mostrato in tabella 4.3.

Classe	Numero elementi
Giunte	853
Marche	743
Ombre	816

Tabella 4.3: *Numero di elementi per ogni classe nel nuovo dataset*

Di questi frame, 300 (100 per ogni classe) sono stati estratti da video differenti rispetto agli altri e sono stati destinati a far parte del test set, per giudicare il comportamento della rete su immagini provenienti da video non noti alla rete. I rimanenti dati sono stati divisi in training set (80%) e validation set (20%), come mostrato in tabella 4.4.

Per creare i tre set si è cercato di utilizzare il più possibile video diversi tra loro, con l'obiettivo di creare un dataset rappresentativo della realtà in esame.

Training set	**Validation set**	**Test set**	**Totale elementi**
1717	430	300	2447

Tabella 4.4: *Divisione degli elementi del nuovo dataset in training, validation e test set*

Sia i parametri della rete che il pre-processing del dataset sono stati replicati dai precedenti esperimenti.

In quest'ultimo esperimento, l'accuratezza ottenuta sul validation set è del 93%. I risultati ottenuti sul test set sono molto buoni, come riportato in figura 4.13, indicando che la rete riesce ad apprendere meglio la differenza tra le varie classi e che il nuovo dataset è formato da un campione rappresentativo.

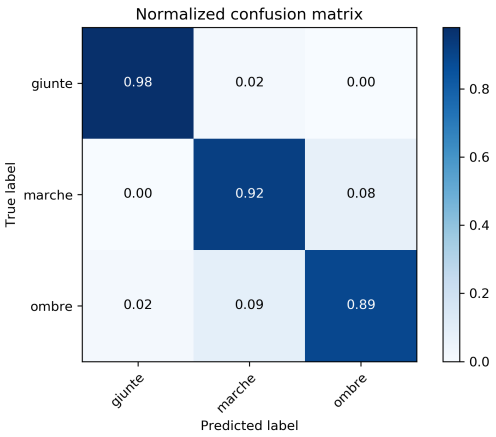


Figura 4.13: *Matrice di confusione ottenuta allenando la rete con il nuovo dataset*

Capitolo 5

Conclusioni e sviluppi futuri

L'evoluzione tecnologica e l'avvento dell'era digitale hanno portato grandi progressi nell'ambito del restauro e della conservazione dei documenti sonori storici. Gli studi di settore hanno così potuto definire delle metodologie sistematiche per le operazioni di *rimediazione* e generare efficientemente sia copie conservative sia copie d'accesso. In questo contesto, applicazioni sviluppate *ad hoc* possono offrire un valido supporto agli operatori nelle procedure, agevolando le operazioni più ripetitive e inclini ad errori.

Presso il Centro di Sonologia Computazionale dell'Università di Padova è stata sviluppata una prima versione di un software per l'estrazione e la classificazione automatica delle discontinuità presenti sui nastri magnetici.

Tale versione del progetto, tuttavia, presentava alcune criticità: si è quindi condotto uno studio del codice e della logica dell'applicazione in modo tale da intervenire al fine di migliorarla.

Il lavoro è stato suddiviso in due fasi: dapprima si è analizzato e migliorato l'applicativo per l'estrazione dei frame dai video e in un secondo momento si è implementata una seconda rete neurale per classificare i frame del dataset esistente, nel tentativo di ottimizzare i risultati ottenuti dal classificatore precedentemente implementato.

Per la parte di estrazione dei frame si sono ampliate le funzionalità del software, come descritto in sezione 3.2, migliorando conseguentemente sia le prestazioni che la precisione dell'algoritmo.

Per la parte di classificazione, invece, si è scelto di implementare nuovamente la rete GoogLeNet usando nuove tecnologie, così come spiegato nel capitolo 4. Testando la rete su immagini nuove e provenienti da video sconosciuti alla rete, è però emerso che il dataset non è rappresentativo della realtà in esame e la rete quindi stenta a generalizzare i concetti e, di conseguenza, a imparare. È notevole, infatti, la divergenza dei risultati della predizione su un campione prelevato in maniera random dal dataset (accuratezza del 99%) e quelli ottenuti su un test set formato da frame estratti da nuovi video (accuratezza del 60%). In un primo momento si sono implementate delle strategie per migliorare i risultati sul dataset esistente, per giudicare come avrebbero reagito i diversi classificatori: sono state aggiunte nuove immagini in fase di allenamento e si sono trasformati i frame in scala di grigi. In tal modo la rete generalizza maggiormente, ottenendo risultati leggermente migliori, ma ancora poco soddisfacenti. In un secondo momento si è deciso

di analizzare il dataset e studiare quali fossero le problematiche.

I problemi principali che sono emersi sono due:

- le immagini non sono costituite dal solo nastro e dalle discontinuità che vi scorrono; di conseguenza, presentano molti elementi che possono confondere la rete, la quale estrae *feature* che non riguardano la specifica discontinuità;
- i frame che compongono il dataset sono troppo simili, la maggior parte sono stati estratti dagli stessi video; la rete in questo caso apprende bene i casi presenti nel dataset ma non riesce a generalizzare (*overfitting*).

Di conseguenza, è nata la necessità di ricostruire l'intero dataset. Dall'analisi di diversi video sono stati raccolti i sotto-frame dell'area relativa al solo nastro. Questo nuovo dataset ha condotto a risultati migliori in fase di test su un campione di immagini estratte da video non noti alla rete.

Sviluppi futuri

Tale progetto dovrà essere sviluppato ulteriormente e, in particolare, offre numerosi spunti per la parte di classificazione.

Sarà innanzitutto necessario analizzare nuovi video, allo scopo di aggiungere nuove immagini al dataset per renderlo più solido. Dopo una fase di miglioramento dei risultati sulle quattro classi prese in considerazione per questo primo esperimento col nuovo dataset, si potrà considerare di aggiungere le classi `rovinato` e `segni`, a patto di raccogliere abbastanza campioni affinchè la rete apprenda in modo esauriente.

Una volta ottenuto un dataset affidabile, sarà possibile implementare nuove reti per confrontare i risultati rispetto alla GoogLeNet.

Una tipologia di reti che si sta diffondendo per la classificazione di immagini è la ResNet (*Residual neural Network*), vincitrice della competizione ILSVRC nel 2015 [10].

La novità principale introdotta da questa rete è il blocco residuo (figura 5.1), nel quale gli

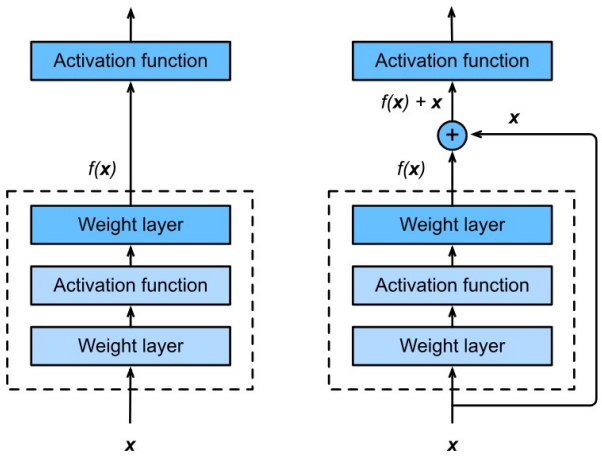


Figura 5.1: *Confronto tra un blocco regolare (sinistra) e un blocco residuo (destra)*

input di un livello inferiore sono resi disponibili per un nodo in uno strato superiore, venendo sommati all'output del blocco inferiore.

In futuro sarà possibile, inoltre, integrare altri progetti sviluppati all'interno del laboratorio e unificare quindi tutte le ricerche portate avanti nell'ambito della conservazione e dell'accesso alle opere sonore. In particolare:

- la fase di estrazione dei frame più importanti può essere integrata nell'applicazione REMIND (sezione 2.1), in modo da selezionare la ROI direttamente sul tablet.
- sia l'estrazione delle discontinuità sia la loro classificazione possono essere integrate con l'interfaccia d'accesso sviluppata presso il CSC (sezione 2.1), in modo da inserire automaticamente i POI video.

Appendici

Appendice A

Estrazione frame

✓ : evento rilevato

× : evento non rilevato

- : il nastro non presenta marche

Video	**Durata analisi**	**Durata video**	**Fine nastro**	**Prima marca**
42a	20:25	1:06:13	✓	-
44	2:30	7:27	✓	×
49	6:21	21:09	✓	✓
55	6:21	19:42	✓	✓
57	6.55	21:26	✓	✓
58	5:08	17:16	✓	-
61a_15	5:40	15:29	✓	✓
62	2:23	8:13	×	✓
63a_15	2:43	8:58	✓	✓
63a_712	5:24	17:50	✓	✓
63b_15	2:41	8:58	✓	✓
61a_712	9:39	30:54	✓	-
69	3:38	12:17	✓	×
70a_15	5:09	16:52	✓	✓
72	4:24	14:29	✓	✓
73	9:13	30:28	✓	-
75	4:22	14:32	✓	✓
76	4:14	14:32	✓	×
79	5.16	17:12	✓	-
80	4:34	15:33	×	✓
82a_712	8:17	27:10	✓	-
82b_15	4:10	13:39	✓	-
82b_712	8:12	27:06	×	-

300	4:36	15:35	✓	-
301	9:53	32:56	✓	-
302	9:50	33:22	✗	-
303	6:46	22:27	✓	-
304	4:28	15:15	✓	-
305	6:55	23:14	✓	✓
306	4:41	16:01	✓	-
307	8:15	28:32	✓	✓
308	8:00	27:42	✓	-
309	8:36	28:59	✓	✓
310	0:52	2:50	✓	-
311	0:33	2:09	✓	-
312	6:33	23:11	✓	-
313	1:50	6:15	✓	-
314	4:30	15:17	✓	✗
315	7:32	25:03	✓	-
316	4:38	15:36	✓	✓
317	2:20	7:36	✓	-
318	4:54	16:33	✓	-
320	10:00	34:02	✓	-
321	10:12	33:54	✓	-
322	3:15	10:15	✓	-
323	2:50	9:50	✗	-
324	16:30	57:32	✓	-
325	1:17	4:21	✗	-
326	13:11	46:12	✓	-
327	14:52	51:48	✓	-
333	1:11	4:02	✓	-
338	2:17	7:40	✓	-
339	8:23	28:27	✓	-
342	8:59	30:29	✓	-
343	25:54	1:27:39	✓	-
344	9:10	30:47	✓	-
346	9:32	31:13	✓	-
347	17:28	54:00	✓	✓
347_A	15:51	55:51	✓	✓
347_CFR	16:23	55:50	✓	-
350a	8:25	30:23	✓	-
350b	8:28	30:35	✓	-
353b	9:20	48:41	✓	-
354	9:43	27:40	✓	-
356	9:58	33:24	✓	-

361a	15:22	48:18	✓	-
362	9:21	23:16	✓	×
363	4:10	13:33	✓	×
364	9:16	31:01	✓	-
365	11:37	32:50	✓	-
368	4:20	12:50	✓	-
375	7:32	23:33	✓	-

Tabella A.1: *Statistiche dell'algoritmo di estrazione*

Appendice B

Implementazione GoogLeNet

```
#definisco modulo Inception
def inception_module(x,
                filters_1x1,
                filters_3x3_reduce,
                filters_3x3,
                filters_5x5_reduce,
                filters_5x5,
                filters_pool_proj,
                name=None):
   conv_1x1 = Conv2D(filters_1x1, (1,1), padding='same',
      activation='relu', kernel_initializer=kernel_init,
      bias_initializer=bias_init)(x)
   conv_3x3 = Conv2D(filters_3x3_reduce, (1, 1), padding='same'
      , activation='relu', kernel_initializer=kernel_init,
      bias_initializer=bias_init)(x)
   conv_3x3 = Conv2D(filters_3x3, (3, 3), padding='same',
      activation='relu', kernel_initializer=kernel_init,
      bias_initializer=bias_init)(conv_3x3)
   conv_5x5 = Conv2D(filters_5x5_reduce,(1, 1),padding='same',
      activation='relu',kernel_initializer=kernel_init,
      bias_initializer=bias_init)(x)
   conv_5x5 = Conv2D(filters_5x5, (5, 5), padding='same',
      activation='relu', kernel_initializer=kernel_init,
      bias_initializer=bias_init)(conv_5x5)
   pool_proj = MaxPool2D((3,3), strides=(1,1), padding='same')(
      x)
   pool_proj = Conv2D(filters_pool_proj, (1, 1), padding='same'
      , activation='relu', kernel_initializer=kernel_init,
      bias_initializer=bias_init)(pool_proj)
```

```
    output = concatenate([conv_1x1, conv_3x3, conv_5x5,
       pool_proj], axis=3, name=name)
    return output

#Creo input layer
input_layer = Input(shape=(224, 224, 3))

#Creo primo blocco della rete neurale: Stem
stem = Conv2D(64, (7, 7), padding='same', strides=(2, 2),
   activation='relu', name='conv_1_7x7/2', kernel_initializer=
   kernel_init, bias_initializer=bias_init)(input_layer)
stem = MaxPool2D((3, 3), padding='same', strides=(2, 2), name='
   max_pool_1_3x3/2')(stem)
stem = Conv2D(64, (1, 1), padding='same', strides=(1, 1),
   activation='relu', name='conv_2a_3x3/1')(stem)
stem = Conv2D(192, (3, 3), padding='same', strides=(1, 1),
   activation='relu', name='conv_2b_3x3/1')(stem)
stem = MaxPool2D((3, 3), padding='same', strides=(2, 2), name='
   max_pool_2_3x3/2')(stem)

x = inception_module(stem, filters_1x1=64, filters_3x3_reduce
   =96, filters_3x3=128, filters_5x5_reduce=16, filters_5x5=32,
   filters_pool_proj=32, name='inception_3a')
x = inception_module(x, filters_1x1=128, filters_3x3_reduce
   =128, filters_3x3=192, filters_5x5_reduce=32, filters_5x5=96,
    filters_pool_proj=64, name='inception_3b')

x = MaxPool2D((3, 3), padding='same', strides=(2, 2), name='
   max_pool_3_3x3/2')(x)

x = inception_module(x, filters_1x1=192, filters_3x3_reduce=96,
    filters_3x3=208, filters_5x5_reduce=16, filters_5x5=48,
   filters_pool_proj=64, name='inception_4a')

#primo ausiliary output
x1 = AveragePooling2D((5, 5), strides=3)(x)
x1 = Conv2D(128, (1, 1), padding='same', activation='relu')(x1)
x1 = Flatten()(x1)
x1 = Dense(1024, activation='relu')(x1)
x1 = Dropout(0.7)(x1) #Tecnica di regolarizzazione: Dropout,
   per prevenire overfitting
```

```
x1 = Dense(6,activation='softmax',name='auxilliary_output_1')(
   x1)

x = inception_module(x, filters_1x1=160, filters_3x3_reduce
   =112, filters_3x3=224, filters_5x5_reduce=24, filters_5x5=64,
    filters_pool_proj=64, name='inception_4b')
x = inception_module(x, filters_1x1=128, filters_3x3_reduce
   =128, filters_3x3=256, filters_5x5_reduce=24, filters_5x5=64,
    filters_pool_proj=64, name='inception_4c')
x = inception_module(x, filters_1x1=112, filters_3x3_reduce
   =144, filters_3x3=288, filters_5x5_reduce=32, filters_5x5=64,
    filters_pool_proj=64, name='inception_4d')

#secondo ausiliary output
x2 = AveragePooling2D((5, 5), strides=3)(x)
x2 = Conv2D(128, (1, 1), padding='same', activation='relu')(x2)
x2 = Flatten()(x2)
x2 = Dense(1024, activation='relu')(x2)
x2 = Dropout(0.7)(x2)
x2 = Dense(6, activation='softmax', name='auxilliary_output_2')
   (x2)

x = inception_module(x, filters_1x1=256, filters_3x3_reduce
   =160, filters_3x3=320, filters_5x5_reduce=32, filters_5x5
   =128, filters_pool_proj=128, name='inception_4e')
x = MaxPool2D((3, 3), padding='same', strides=(2, 2), name='
   max_pool_4_3x3/2')(x)
x = inception_module(x, filters_1x1=256, filters_3x3_reduce
   =160, filters_3x3=320, filters_5x5_reduce=32, filters_5x5
   =128, filters_pool_proj=128, name='inception_5a')
x = inception_module(x, filters_1x1=384, filters_3x3_reduce
   =192, filters_3x3=384, filters_5x5_reduce=48, filters_5x5
   =128, filters_pool_proj=128, name='inception_5b')

#output layer
out = GlobalAveragePooling2D(name='avg_pool_5_3x3/1')(x)
out = Dropout(0.4)(out)
out = Dense(6, activation='softmax', name='output')(out)

#creazione modello
model = Model(input_layer,[out, x1, x2],name='googleNetKeras')
```

Appendice C

Gestione ROI

Selezione ROI

Di seguito è riportato un estratto di codice della funzione per la gestione dell'input del mouse per la selezione della ROI: `void mouseHandler(int event, int x, int y, int flags, void* param)`.

```
if(first || capstainSelection){ //creo primo rettangolo
  if (event == CV_EVENT_LBUTTONDOWN && !drag && !select_flag) {
    /* Click con pulsante sx. Inizia la selezione della ROI */
    point1 = cv::Point(x, y);
    drag = 1;
    if (capstainSelection) {
      xc_l = x;
      yc_u = y;
    }
    else
    { x_l = x; y_u = y; }
  }
  if (event == CV_EVENT_MOUSEMOVE && drag && !select_flag) {
    /* Mouse in movimento */
    cv::Mat img1 = myframe.clone();
    point2 = cv::Point(x, y);
    cv::rectangle(img1, point1, point2, cv::Scalar(255,255,0),
       2);
    cv::imshow(src_window, img1);
  }
  if (event == CV_EVENT_LBUTTONUP && drag && !select_flag) {
    /* Pulsante sx rilasciato. Fine selezione ROI */
    cv::Mat img2 = myframe.clone();
    point2 = cv::Point(x, y);
```

```
    drag = 0;
    cv::rectangle(img2, point1, point2, cv::Scalar(255,255,0),
       2);
    cv::imshow(src_window, img2);
    callback = true;
    first=0;
    if (capstanSelection) { //selezione ROI per capstan
      xc_r = x;
      yc_d = y;
      capstainSelection = 0;
      capstainRect=1;
    }
    else { // selezione ROI per nastro
      x_r = x;
      y_d = y;
    }
  }
}
```

Rotazione ROI

Estratto di codice per la gestione della rotazione del rettangolo:

```
if(flags==(EVENT_FLAG_CTRLKEY)) {
  if (event == CV_EVENT_LBUTTONDOWN && !drag && !select_flag) {
    point1 = cv::Point(x,y);
    //Trovo il centro del rettangolo
    xM = (x_l+x_r)/2;
    yM = (y_u+y_d)/2;
    M = cv::Point(xM,yM);
    drag = 1;
    rotating=1;
  }
  if (event == CV_EVENT_MOUSEMOVE && drag && !select_flag) {
    /*  Mouse in movimento, mostro all'utente il rettangolo man
       mano che viene ruotato
      Codice simile a quanto riportato sotto */
  }
  if (event == CV_EVENT_LBUTTONUP && drag && !select_flag) {
    drag=0;
    cv::Mat img2 = myframe.clone();
    point2 = cv::Point(x,y);
```

```
RotatedRect rRect = RotatedRect(Point2f(xM,yM),
            Size2f(x_r - x_l,y_d - y_u),
            point2.x-point1.x);
Point2f vertices[4];
rRect.points(vertices);
for (int i = 0; i < 4; i++)
  cv::line(img2, vertices[i], vertices[(i+1)%4],
     Scalar(255,255,0), 2);
cv::imshow(src_window, img2);

double m1, q1, m2, q2, m3, q3, m4, q4;
/* Trovo le rette che delimitano il nuovo rettangolo */
findRectBound(vertices[0],vertices[1],vertices[2],vertices
   [3],
      m1, m2, m3, m4, q1, q2, q3, q4);

// Trovo ordine dei vertici dall'alto verso il basso
int ordinate[4]= {(int)vertices[0].y,(int)vertices[1].y,
        (int)vertices[2].y,(int)vertices[3].y};
std::sort(ordinate, ordinate+4);

if(*std::min_element(ordinate,ordinate+4)==(int)vertices[2].
   y){
  //y2 primo
  y_u=vertices[2].y;
  y_d=vertices[0].y;
  mA = m3; qA = q3; mD = m2; qD = q2;
  mB = m4; qB = q4; mC = m1; qC = q1;
  if(*std::min_element(ordinate+1,ordinate+3)==(int)vertices
     [3].y){
    y_m1 = vertices[3].y; //y3 secondo
    y_m2 = vertices[1].y; //y1 terzo
  }
  else { //caso frequente nei video
    y_m1 = vertices[1].y; //y1
    y_m2 = vertices[3].y; //y3
  }
}
else if(*std::min_element(ordinate,ordinate+4)==(int)
   vertices[0].y)
{ /* ordine: y0,y1,y3,y2 o y0,y3,y1,y2 */ }
```

```
    else if(*std::min_element(ordinate,ordinate+4)==(int)
       vertices[1].y)
    { /*ordine: y1,y2,y0,y3 o y1,y0,y2,y3*/ }
    else
    { /*ordine: y3,y0,y2,y1 o y3,y2,y0,y1*/ }

    float ascisse [4] = {vertices[0].x, vertices[1].x,
                  vertices[2].x,vertices[3].x};
    float min_ascissa=ascisse[4];
    float max_ascissa=ascisse[0];
    for(int j=0; j<4; j++) {
      if (ascisse[j]>max_ascissa)
      { max_ascissa=ascisse[j]; }
      if (ascisse[j]<min_ascissa)
      { min_ascissa=ascisse[j]; }
    }
    x_r=max_ascissa;
    x_l=min_ascissa;
    rotating=0;
  }
}
```

Modifica ROI

Estratto di codice per la gestione della modifica del rettangolo:

```
if (event == CV_EVENT_MOUSEMOVE && drag && !select_flag && !
   first){
  //mentre il mouse si muove mostro all'utente il nuovo
     rettangolo
  cv::Mat img4 = myframe.clone();
  point4 = cv::Point(x, y);
  point2b = cv::Point(point4.x, y_d); //x mouse, y_d
  point1b = cv::Point(point4.x, y_u); //x mouse, y_u
  point1c = cv::Point(x_l, point4.y); //x_l, y mouse
  point2c = cv::Point(x_r, point4.y); //x_r, y mouse

  //scegliere quale lato muovere se sono dentro al rettangolo
  if(point3.x >= (x_l+x_r)/2) {//click nell'area dx del
     rettangolo
    if (std::abs(point3.y-y_u) >= std::abs(point3.y-y_d)){
      if(std::abs(point3.y-y_d) <= std::abs(point3.x-x_r)){
```

```
      //Devo spostare lato inferiore
      cv::rectangle(img4,point1,point2c,cv::Scalar(255,0,0),2);
      cv::imshow(src_window, img4);
      point2=point2c;
     }
     else { //Devo spostare lato destro
      cv::rectangle(img4,point1,point2b,cv::Scalar(255,0,0),2);
      cv::imshow(src_window, img4);
      point2=point2b;
     }
    }
    else {
     if (std::abs(point3.y-y_u) <= std::abs(point3.x-x_r)){
      //Devo spostare lato sopra
      cv::rectangle(img4,point1c,point2,cv::Scalar(255,0,0),2);
      cv::imshow(src_window, img4);
      point1=point1c;
     }
     else {//Devo spostare lato destro
      cv::rectangle(img4,point1,point2b,cv::Scalar(255,0,0),2);
      cv::imshow(src_window, img4);
      point2=point2b;
     }
    }
  }
  else { /*Codice per la gestione dell'area sx del rettangolo */
      }
}
if (event == CV_EVENT_LBUTTONUP && drag && !select_flag) {
  /*Quando l'utente rilascia il mouse, salvo i nuovi lati del
     rettangolo*/
  cv::Mat img5 = myframe.clone();
  point5 = cv::Point(x, y);
  drag = 0;
  point2b = cv::Point(point5.x, y_d);
  point1b = cv::Point(point5.x, y_u);
  point1c = cv::Point(x_l, point5.y);
  point2c = cv::Point(x_r, point5.y);

  if(point3.x >= (x_l+x_r)/2) { //click nell'area dx del
     rettangolo
    if (std::abs(point3.y-y_u)>=std::abs(point3.y-y_d)){
```

```
    if(std::abs(point3.y-y_d)<=std::abs(point3.x-x_r)){
      //sposto lato inferiore
      cv::rectangle(img5, point1, point2c, cv::Scalar(255,0,0),
          2);
      cv::imshow(src_window, img5);
      point2=point2c;
      y_d = point2.y;
    }
    else { //Lato destro
      cv::rectangle(img5, point1, point2b, cv::Scalar(255,0,0),
          2);
      cv::imshow(src_window, img5);
      point2=point2b;
      x_r = point2.x;
    }
  }
  else {
    if (std::abs(point3.y-y_u) <= std::abs(point3.x-x_r)) {
      //Sposto lato superiore
      cv::rectangle(img5, point1c, point2, cv::Scalar(255,0,0),
          2);
      cv::imshow(src_window, img5);
      point1=point1c;
      y_u = point1.y;
    }
    else { //Lato destro
      cv::rectangle(img5, point1, point2b, cv::Scalar(255,0,0),
          2);
      cv::imshow(src_window, img5);
      point2=point2b;
      x_r = point2.x;
    }
  }
 }
 else //sono nel lato sx
 { /*Codice per gestione dell'area sx del rettangolo */ }
}
```

Bibliografia

[1] Federica Bressan and Sergio Canazza. A systemic approach to the preservation of audio documents: Methodology and software tools. *Journal of Electrical and Computer Engineering*, 2013.

[2] Federica Bressan, Sergio Canazza, Niccolò Pretto, and Carlo Fantozzi. Tape music archives: from preservation to access. *International Journal on Digital Libraries*, 2017.

[3] S. Canazza, G. De Poli, and A. Vidolin. La conservazione dei documenti audio: un'innovazione in prospettiva storica. *Archivi, VI (2)*, pages 7–56, 2011.

[4] Daniele Colanardi. Definizione di un modello e sviluppo di un framework per l'archiviazione e il supporto all'analisi musicologica di documenti sonori storici. Tesi di Laurea Magistrale, Università degli Studi di Padova, 2017.

[5] IASA Technical Committee. The safe guarding of the audio heritage: Ethics, principles and preservation strategy, December 2005.

[6] Ray Edmondson. *Memory of the world: general guidelines to safeguard ducumentary heritage*. UNESCO, 2002.

[7] Susan Eldridge. Digital audio tapes: Their preservation and conversion. *Smithsonian Institution Archives*, 2010.

[8] Ylenia Grava. Progettazione e sviluppo di un'applicazione web-based per l'archiviazione e il supporto all'analisi musicologica di documenti sonori storici. Tesi di Laurea, Università degli Studi di Padova, 2017.

[9] IASA Editorial Group. *The IASA Cataloguing Rules*. International Association of Sound and Audiovisual Archives, 1999.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[11] IFLA/UNESCO. *Safeguarding our Documentary Heritage/Conservation préventive du patrimoine documentaire*. French Ministry of Culture and Communication, Boston, Massachussetts, USA, 2000.

[12] Mirco Maniero. Analisi automatica di nastri magnetici con reti neurali e tecniche di visione computazionale. Tesi di Laurea Magistrale, Università degli Studi di Padova, 2016.

[13] Robin Pike, Kathlin Smith, Sam Brylawski, and Maya Lerman. Arsc guide to audio preservation, 2015.

[14] Niccolò Pretto. *Cultural context-aware models and IT applications for the exploitation of musical heritage*. PhD thesis, Università degli Studi di Padova, 2019.

[15] Niccolò Pretto, Carlo Fantozzi, Edoardo Micheloni, Valentina Burini, and Sergio Canazza. Computing methodologies supporting automatic analysis of electroacoustic music on analog magnetic tape. *Computer Music Journal*, 42(4), 2018.

[16] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.