

Università degli Studi di Modena e Reggio Emilia

DIPARTIMENTO DI INGEGNERIA “ENZO FERRARI”

Corso di Laurea Magistrale in Ingegneria Informatica

**Riconoscimento di gesti per l'interazione
uomo-veicolo mediante sensori infrarossi e 3D**

Candidato:

Fabio Manganaro

Relatore:

Prof. Roberto Vezzani

Correlatori:

Dott. Guido Borghi

Dott. Stefano Pini

Abstract

La distrazione del guidatore è una delle tematiche principali affrontate nel settore *Automotive* da diverse comunità, compresa quella di visione artificiale. L'Eu-ro NCAP (Programma europeo di valutazione dei nuovi modelli di automobili) pianifica di introdurre obbligatoriamente dal 2020 diversi sistemi per il monito-raggio dello stato del conducente all'interno dei veicoli.

Una delle principali fonti di distrazione primaria per il *driver* è la sempre più complessa modalità di interazione con il proprio veicolo e i suoi dispositivi, quali il sistema di *infotainment*.

In questa tesi sono stati studiati metodi ed algoritmi per il riconoscimento au-tomatico di gesti tramite immagini acquisite da sensori infrarossi e depth. Le tecniche proposte hanno il duplice obiettivo di semplificare l'interazione uomo-macchina (HMI) e abilitarne la supervisione ed il controllo.

Il sistema proposto si basa su algoritmi di visione artificiale, come reti neurali per l'analisi di immagini, i quali richiedono una grande mole di dati per essere implementati. Per ovviare a ciò, come parte integrante del lavoro di tesi, si è acquisito e rilasciato un *dataset* di gesti realistici, al fine di realizzare un sistema che limiti il più possibile le interazioni tra guidatore e veicolo e garantire una maggiore sicurezza.

Indice

Elenco delle figure	vii
Elenco delle tabelle	xi
1 Introduzione	1
1.1 Automotive HMI	2
1.1.1 Natural User Interface (NUI)	3
1.2 Scopo della tesi	4
1.3 Struttura della tesi	6
2 Gesture Recognition: stato dell'arte	7
2.1 Vision Based Gesture Recognition	8
2.2 Dynamic Hand Gestures Recognition: stato dell'arte	9
2.2.1 VIVA Hand Gesture Dataset	9
2.2.2 NVIDIA Dynamic Hand Gesture Dataset	10
2.2.3 Leap Motion Dynamic Hand Gesture (LMDHG)	11
2.3 Considerazioni finali	11
3 Acquisizione dati	13
3.1 Deep Learning e limiti	13
3.2 Contesto applicativo	14
3.3 Progettazione hardware	14
3.3.1 Configurazione in veicolo	14
3.3.2 Sensori	16
3.3.3 Setup di acquisizione	23
3.4 Gesti Dinamici	23
3.5 Software di acquisizione	25
3.6 Procedura di acquisizione	26
3.7 Conclusioni	27

4 Metodo	29
4.1 Approccio con Reti Neurali Convolute	29
4.1.1 DenseNet	30
4.1.2 Metodo proposto	31
4.2 Approccio con Architettura Multimodale	33
4.2.1 Metodo proposto	33
4.3 Approccio con Reti Neurali Convolute 3D	35
4.3.1 C3D	35
4.3.2 Metodo proposto	36
4.4 Approccio con Reti Neurali Ricorrenti	37
4.4.1 LSTM	37
4.4.2 Metodo proposto	38
5 Risultati sperimentali	41
5.1 Metriche	41
5.2 Approccio con Reti Neurali Convolute	42
5.2.1 Modalità di Training	42
5.2.2 Risultati	45
5.2.3 Analisi Risultati	49
5.3 Approccio con Architettura Multimodale	50
5.3.1 Risultati	50
5.3.2 Analisi Risultati	57
5.4 Approccio con Reti Neurali Convolute 3D	58
5.4.1 Modalità di Training	58
5.4.2 Risultati	61
5.4.3 Analisi Risultati	65
5.5 Approccio con Reti Neurali Ricorrenti	66
5.5.1 Modalità di Training	66
5.5.2 Risultati	67
5.5.3 Analisi Risultati	69
5.6 Conclusioni	69
6 Prototipo di sistema	71
7 Conclusioni	75
8 Ringraziamenti	77
Bibliografia	79

Elenco delle figure

2.1	Rappresentazione grafica dei vari modelli	8
2.2	Esempio di dati provenienti dal VIVA <i>dataset</i> [17]	10
2.3	Ambiente di acquisizione dell’NVIDIA <i>dataset</i> ; dispositivi utilizzati ((A) RGB e depth camera, (B) camera per immagini IR-stereo; modalità di acquisizione (RGB, optical flow, profondità, IR-sinistra, e disparità).	10
2.4	Tipologie di acquisizione - LMDHG	11
3.1	Possibili posizionamenti dei sensori in abitacolo: (a), (b)	15
3.2	Tipica struttura per scanner 3D a luce strutturata	17
3.3	Funzionamento di un classico sensore ToF, lo strumento è composto da due parti, una che si occupa di proiettare il fascio di luce e l’altra di ricevere il fascio riflesso.	18
3.4	Sensore Pico FLeXX	18
3.5	Spettro infrarosso	19
3.6	Esempio di immagini acquisite in bande differenti dell’IR	19
3.7	Sensore RGB.	20
3.8	Immagini acquisite nel <i>dataset</i> : infrarossi, RGB, <i>depth maps</i>	20
3.9	Leap motion e sistema di riferimento.	21
3.10	Giunti mano: coordinate palmo, punta delle dita, ossa.	22
3.11	Immagini acquisite dal Leap Motion: immagine telecamera destra distorta e rettificata, immagine telecamera sinistra distorta e rettificata.	22
3.12	Configurazione plancia e disposizione sensori.	23
3.13	Gesti dinamici: <i>frame</i> IR di preparazione, esecuzione e termine.	25
3.14	Software di acquisizione: schermata per il controllo della procedura ed esempio di immagini acquisite in tempo reale per verificare la corretta acquisizione.	26
3.15	Sessione di registrazione di un soggetto.	27

4.1	Un <i>Dense block</i> della rete con 5 <i>layer</i>	30
4.2	Una <i>deep DenseNet</i> con tre <i>dense block</i>	31
4.3	Stack di N frame. Esempio con immagini infrarossi mono canale.	32
4.4	Architettura Multimodale.	33
4.5	Esempio di Architettura Multimodale utilizzando immagini IR, RGB e Depth Map.	34
4.6	Operazioni di convoluzione 2D e 3D. a) L'applicazione della convoluzione 2D su un'immagine produce un'immagine. b) L'Applicazione della convoluzione 2D su un volume video (più fotogrammi come più canali (vedi 4.1)) genera un'immagine. c) L'applicazione della convoluzione 3D su un volume video produce un altro volume, preservando le informazioni temporali del segnale di ingresso [35].	36
4.7	Architettura C3D. [35].	36
4.8	Architettura di una rete LSTM. X_t rappresenta l'input temporale, h_t l' <i>output (hidden state)</i> prodotto.	38
4.9	Architettura della rete LSTM implementata. h_i tensore i -esimo di input con t variabile (40, 35, 30), l_i livello della rete, A cella lstm, e y_i classe predetta.	39
5.1	Input della rete: depth map. Grafico di accuracy e loss in <i>training</i> e <i>validation</i> per N frame diversi: in verde $\mathcal{N} = 40$; in grigio $\mathcal{N} = 35$; in arancio $\mathcal{N} = 30$	43
5.2	Input della rete: immagini IR. Grafico di accuracy e loss in <i>training</i> e <i>validation</i> per N frame diversi: in blu $\mathcal{N} = 40$; in azzurro $\mathcal{N} = 35$; in verde $\mathcal{N} = 30$	43
5.3	Input della rete: immagini RGB. Grafico di accuracy e loss in <i>training</i> e <i>validation</i> per N frame diversi: in grigio $\mathcal{N} = 40$; in arancio $\mathcal{N} = 35$; in rosso $\mathcal{N} = 30$	44
5.4	Input della rete: immagini RGB in scala di grigi. Grafico di accuracy e loss in <i>training</i> e <i>validation</i> per N frame diversi: in blu $\mathcal{N} = 40$; in azzurro $\mathcal{N} = 35$; in verde $\mathcal{N} = 30$	44
5.5	Matrice di confusione per il tipo di input: depth map. Lunghezza sequenza = 40 frame	46
5.6	Matrice di confusione per il tipo di input: immagini IR. Lunghezza sequenza = 30 frame	47
5.7	Matrice di confusione per il tipo di input: immagini RGB. Lunghezza sequenza = 35 frame	47
5.8	Matrice di confusione per il tipo di input: immagini RGB grayscale. Lunghezza sequenza = 40 frame	48

5.9	Matrice di confusione per il tipo di input: depth map - IR.	51
5.10	Matrice di confusione per il tipo di input: depth map - RGB.	52
5.11	Matrice di confusione per il tipo di input: depth map - RGB grayscale.	53
5.12	Matrice di confusione per il tipo di input: IR - RGB.	53
5.13	Matrice di confusione per il tipo di input: IR - RGB Grayscale. .	54
5.14	Matrice di confusione per il tipo di input: depth map - IR - RGB. .	55
5.15	Matrice di confusione per il tipo di input: depth map - IR - RGB gs.	56
5.16	Input della rete: depth map. Grafico di accuracy e loss in <i>training</i> e <i>validation</i>	59
5.17	Input della rete: immagini IR. Grafico di accuracy e loss in <i>training</i> e <i>validation</i>	59
5.18	Input della rete: immagini RGB. Grafico di accuracy e loss in <i>training</i> e <i>validation</i>	60
5.19	Input della rete: immagini RGB grayscale. Grafico di accuracy e loss in <i>training</i> e <i>validation</i>	60
5.20	Matrice di confusione per il tipo di input: depth map.	62
5.21	Matrice di confusione per il tipo di input: immagini IR.	63
5.22	Matrice di confusione per il tipo di input: immagini RGB.	63
5.23	Matrice di confusione per il tipo di input: immagini RGB grayscale. .	64
5.24	Coordinate 3D dei giunti: grafico di accuracy e loss in <i>training</i> e <i>validation</i> per N frame diversi: in arancio $\mathcal{N} = 40$; in verde $\mathcal{N} = 35$; in grigio $\mathcal{N} = 30$	66
5.25	Matrice di confusione della rete LSTM Lunghezza sequenza = 40 frame.	68
6.1	Diagramma di flusso del sistema prototipale.	73
6.2	Esempio di funzionamento del sistema prototipale: riconoscimento del gesto g00 - <i>fist</i>	73

Elenco delle tabelle

5.1 Accuracy del metodo CNN al variare del tipo di input utilizzato e della lunghezza in frame della sequenza. In grassetto è evidenziato il risultato migliore.	45
5.2 Accuracy per classe per il tipo di input depth map e lunghezza sequenza = 40 frame.	45
5.3 Accuracy per classe per il tipo di input immagini IR e lunghezza sequenza = 30 frame.	46
5.4 Accuracy per classe per il tipo di input immagini RGB e lunghezza sequenza = 35 frame.	46
5.5 Accuracy per classe per il tipo di input immagini RGB grayscale e lunghezza sequenza = 40 frame.	48
5.6 <i>Accuracy</i> media dell'Architettura Multimodale per ciascun tipo di input.	50
5.7 Accuracy per classe per il tipo di input depth map - IR.	51
5.8 Accuracy per classe per il tipo di input depth map - RGB.	51
5.9 Accuracy per classe per il tipo di input depth map - RGB grayscale.	52
5.10 Accuracy per classe per il tipo di input IR - RGB.	52
5.11 Accuracy per classe per il tipo di input IR - RGB grayscale.	54
5.12 Accuracy per classe per il tipo di input depth map - IR - RGB.	54
5.13 Accuracy per classe per il tipo di input depth map - IR - RGB grayscale.	55
5.14 <i>Accuracy</i> media della rete C3D per ciascun tipo di input.	61
5.15 Accuracy per classe per il tipo di input depth map.	61
5.16 Accuracy per classe per il tipo di input immagini IR.	62
5.17 Accuracy per classe per il tipo di input immagini RGB.	62
5.18 Accuracy per classe per il tipo di input immagini RGB grayscale.	64
5.19 Accuracy media del metodo LSTM al variare della lunghezza della sequenza.	67

5.20 <i>Accuracy</i> per classe [g0 ... g11] della rete neurale ricorrente (LSTM) per il tipo di input: feature 3D. <i>Accuracy</i> calcolata per sequenze di lunghezza 40.	67
5.21 Accuracy delle configurazioni migliori dei metodi proposti. Modello di rete; tipo di input; lunghezza sequenza in <i>frame</i>	69

Capitolo 1

Introduzione

Più del novanta per cento degli incidenti stradali sono dovuti a *errori umani* [7]. In generale, possono essere osservate due tipologie di cause: *violazioni*, nelle quali rientrano le più comuni infrazioni per eccessiva velocità e guida sotto influenza di alcool o droghe; e *errori umani*, nei quali risultano essere di primaria rilevanza lo stato del conducente - inattività, fatica, distrazione - e l'inesperienza alla guida.

Concentrandoci, dunque, su quest'ultima tipologia di errori, assume particolare importanza definire più dettagliatamente quanti e quali sono gli stati di potenziale pericolo nei quali il *driver* può identificarsi. Nel settore dell'*automotive* si è soliti individuare come rilevanti per la conduzione in sicurezza di un veicolo i seguenti stati del conducente [8]:

- **disattenzione del conducente:** si verifica quando l'allocazione di risorse del driver alle attività inerenti alla guida non corrisponde alle esigenze delle attività richieste per il controllo dei margini di sicurezza;
- **distrazione del conducente:** si verifica quando il driver assegna le proprie risorse ad attività non legate alla sicurezza, mentre le risorse assegnate ad attività fondamentali per una guida sicura non corrispondono a quelle richieste per queste ultime;
- **fatica del conducente:** quando sono presenti livelli alti o bassi di fatica che possono compromettere l'abilità del conducente nel monitorare l'ambiente di guida.

In particolar modo, la distrazione del conducente dalla sua funzione primaria - la guida - risulta essere una delle maggiori cause di incidenti stradali [6]. Dal 2011, circa il 15% degli incidenti verificatisi è imputabile alla distrazione durante la guida. Al giorno d'oggi, il 21% degli scontri frontali è causato dall'utilizzo di smartphone, in concordanza con il dato registrato che vede il 35% dei gui-

datori con età inferiore ai 20 anni fare uso del proprio smartphone in maniera continuativa durante la guida.

L'agenzia *National Highway Traffic Safety Administration* (NHTSA), in accordo con la definizione data precedentemente (vedi 1), identifica come distrazione una qualsiasi attività che può distogliere l'attenzione di una persona dall'attività di guida. Vengono specificati, inoltre, le seguenti tipologie di distrazione [21]:

- **distrazione manuale:** si verifica quando il driver svolge attività che gli impediscono di avere le mani sul volante;
- **distrazione visiva:** si verifica quando il driver volge lo sguardo verso un punto di interesse diverso dalla strada;
- **distrazione cognitiva:** si verifica quando stress e altri fattori relativi costringono il driver a non dedicarsi con adeguata sicurezza alla guida.

Si delinea sempre di più, dunque, la necessità di contrastare con efficacia le numerosi fonti di distrazione che possono minacciare la sicurezza del conducente, e dei relativi passeggeri, nell'utilizzo del proprio veicolo.

Le possibili contromisure adottabili per far fronte a questo elevato tasso di incidenti stradali dovuti alla distrazione del conducente sono:

- Regole e leggi;
- Addestramento del pilota;
- Campagne pubbliche per l'informazione e la prevenzione;
- Adeguamento delle infrastrutture;
- **Strumenti basati su tecnologie.**

Proprio su quest'ultima possibile contromisura si concentra il lavoro svolto e presentato in questa tesi: l'ideazione di un sistema intuitivo ed immediato di interfaccia uomo-veicolo come strumento tecnologico per incrementare la sicurezza durante la guida.

1.1 Automotive HMI

Nell'utilizzo completo del proprio mezzo di trasporto, uno degli elementi principali che caratterizza l'esperienza di guida è l'interazione tra conducente e veicolo. Al giorno d'oggi, essa si esplica nell'utilizzo di sempre più sofisticate interfacce uomo-macchina (HMI).

Tradizionalmente, queste interfacce consistono in sistemi multipli che permettono al guidatore di interagire con il veicolo mediante l'uso di manopole, rotori o display touch screen, e nelle implementazioni di oggi, le HMI forniscono anche numerosi feedback al conducente [3]. L'interazione del conducente col proprio

veicolo comincia nell'istante in cui il driver sblocca la portiera, continua durante la guida, e termina nel momento in cui il guidatore arresta la corsa ed esce dal veicolo. Per avere un'esperienza di guida godibile e soprattutto sicura, è richiesto un bilanciamento ottimale degli input sensoriali, come segnali acustici o visivi, forniti al conducente.

Il modulo HMI principale attraverso il quale il guidatore interagisce con il veicolo per modificarne o visualizzarne le impostazioni, dalla climatizzazione alla stazione radio, è il sistema di *infotainment*, termine che deriva dall'unione delle parole *information* ed *entertainment*. Inoltre, le moderne interfacce sono state estese nelle funzionalità per permettere al guidatore di controllare e accedere a dispositivi elettronici personali come gli smartphone.

L'industria dell'automobile sta attraversando una rivoluzione delle HMI che continua a modificare il modo con cui guidatore e passeggeri interagiscono con i propri veicoli. Negli ultimi dieci anni si sono raggiunte diverse soluzioni innovative per migliorare l'esperienza di guida ed in generale l'utilizzo del veicolo [2]:

- tecnologie avanzate di display, come l'utilizzo di touch screen, hanno modificato il *layout* delle informazioni fornite, da un sistema statico ad uno dinamico e flessibile.
- dispositivi *aptici*, come volanti che vibrano in caso di allerta, sono diventati facilmente disponibili, fornendo nuovi canali per dare al guidatore un feedback sullo stato di guida.

La rivoluzione ci sta dunque portando da interfacce uomo-macchina (HMIInterface) a sempre più complesse interazioni uomo-macchina (HMInteraction). Questa continua evoluzione però, porta con sé anche degli aspetti negativi. La distrazione del conducente, principale causa di incidenti stradali di cui abbiamo precedentemente discusso, risulta essere influenzata negativamente dal progredire in complessità delle interfacce rese disponibili dal veicolo. Risulta necessario dunque trovare un giusto compromesso tra quantità e qualità di informazioni a cui è possibile accedere, e relative modalità di interazione, per non inficiare sul fattore sicurezza.

1.1.1 Natural User Interface (NUI)

Nel tentativo di risolvere il vincolo individuato precedentemente, trovano un giustificato utilizzo le Natural User Interface (NUI), ovvero interfacce che non necessitano di strumenti fisici a disposizione dell'utente per essere controllate. Wigdor e Wixon descrivono una NUI come *un'interfaccia che fa sì che l'utente agisca e si senta naturale* [4]. Ciò non comporta che le interfacce debbano

mimare il mondo reale ma che piuttosto debbano creare un'esperienza che possa essere associata ad un'estensione del loro corpo. Inoltre, le NUI devono risultare godibili e soprattutto, per quanto riguarda il fattore sicurezza, appropriate al contesto di utilizzo.

I tradizionali mezzi di interfaccia in ambito *automotive*, quali rotori, manopole e sempre più diffusi ed ampi *touch screen*, risultano essere, se mal progettati, poco intuitivi ed immediati, inficiando uno dei principi cardine delle *natural user interfaces*, l'usabilità. È proprio attraverso l'utilizzo di interfacce di quest'ultima tipologia che il paradigma dello *user friendly* trova la sua massima espressione, garantendo dunque una più efficiente interazione da parte dell'utente.

Interfacce naturali mediante gesti o riconoscimento del parlato, già esplorate in ambito di ricerca, sono divenute più che mai elementi ricorrenti all'interno del settore dell'automobile, poiché producono un incremento del livello di velocità e quindi una maggiore sicurezza del guidatore nell'interazione con il veicolo. Il guidatore infatti non viene forzato a distogliere la propria attenzione dalla guida per un lasso di tempo rilevante e dunque potenzialmente pericoloso. Inoltre, oggi i conducenti sono più impegnati in compiti secondari rispetto al passato, a causa della presenza, per esempio, di smartphone, che influiscono fortemente sullo stato attentivo del guidatore [10].

L'integrazione delle NUI nel settore *automotive* dunque ha il potenziale di ridurre, in determinati casi, la distrazione del conducente e dar forma a nuove esperienze per interagire con il sistema di *infotainment* rispettando, il più possibile, i vincolanti parametri di sicurezza.

In relazione a quanto detto, la tendenza che il settore *automotive* sta seguendo è quella quindi di mettere in sicurezza il guidatore, e nel contempo, rendere migliore la sua esperienza di guida, tramite interfacce sempre più intuitive, immediate, naturali.

1.2 Scopo della tesi

L'argomento di cui tratterà questo tesi che è stata portata avanti all'interno del RedVision Lab in collaborazione con Ferrari S.p.A., risiede nello studio di un sistema di *natural user interface* attraverso il quale il guidatore potrà interagire con il proprio veicolo: il *riconoscimento di gesti*. Tale sistema, basato su tecnologie di visione, deve essere in grado di riconoscere automaticamente in maniera corretta i gesti che il conducente compie, in modo da interagire in maniera efficiente con il proprio mezzo di trasporto ed in modo specifico con il modulo di *infotainment*.

Durante l'elaborazione del sistema di riconoscimento sono state svolte attività di ideazione, progettazione ed implementazione che possono essere suddivise nei seguenti *step*.

Inizialmente il lavoro si è concentrato sullo studio della tecnologia di visione applicabile all'interno del veicolo e del suo posizionamento (plancia, tunnel centrale, etc.), rispettando i vincoli progettuali posti dal settore *automotive*. Tale studio ha portato all'individuazione di tre tecnologie utilizzabili: sensori RGB, infrarossi (IR) e 3D. Inoltre, di primaria importanza è stato lo studio e l'identificazione di dodici gesti attraverso i quali il guidatore possa interagire con il proprio veicolo.

In seguito, si è svolto uno studio della letteratura sulla situazione tecnologica attuale che permetta il riconoscimento di gesti all'interno di un veicolo. Questa fase ha prodotto come risultato l'individuazione di *reti neurali* applicate al mondo della visione come principale soluzione al problema.

La fase successiva, dovuta principalmente alla soluzione tecnologica individuata e alla mancanza di materiale pubblico apposito per la configurazione tecnologica proposta, dovuta alla posizione e tipologia dei sensori differenti da quelli scelti in fase di progettazione, è stata l'acquisizione di un *dataset* di *hand gesture*, attraverso il quale addestrare il sistema di riconoscimento da implementare.

Di conseguenza si è proceduto con l'implementazione e comparazione di diverse soluzioni di *reti neurali* al fine di identificare la migliore modalità di interazione uomo-veicolo. Fra le tipologie studiate, troviamo le Reti Neurali Convulsive (CNN), le Reti Neurali Convulsive 3D e le Reti Neurali Ricorrenti (RNN).

Infine, è stato realizzato un prototipo funzionante in real time in grado di emulare sistema di riconoscimento di gesti finale, utilizzando il metodo più efficiente precedentemente individuato ed implementato.

1.3 Struttura della tesi

L'elaborato è organizzato come segue:

- il **capitolo 2** espone la situazione attuale del mondo della *gesture recognition*, di come viene affrontata in ricerca, dei vincoli di utilizzo e delle relative soluzioni implementate dalla comunità scientifica;
- il **capitolo 3** tratta della creazione *ad hoc* del *dataset*, del perché è stato necessario acquisirlo, della progettazione e ideazione delle *gesture*, dei requisiti necessari, della tecnologia utilizzata, dei dati ottenuti e della sua struttura finale;
- nel **capitolo 4** vengono affrontate, analizzate e studiate le soluzioni individuate al problema del riconoscimento dei gesti, i metodi e le architetture di reti neurali scelte ed implementate;
- il **capitolo 5** mostra gli esperimenti effettuati, delle modalità con cui le soluzioni architetturali implementate sono state testate ed i relativi risultati ottenuti;
- nel **capitolo 6** viene esposto il prototipo funzionante in real time del sistema di riconoscimento di gesti implementato e del perché delle scelte ingegneristiche effettuate;
- il **capitolo 7** è dedicato alle conclusioni sul lavoro svolto ed ai possibili sviluppi futuri.

Capitolo 2

Gesture Recognition: stato dell'arte

Con *Gesture Recognition* (riconoscimento dei gesti) ci si riferisce al *task* che si pone come obiettivo l'interpretazione dei gesti umani provenienti non solo dalle mani, ma anche dall'intero corpo.

Qualunque sistema basato su questa tipologia di interfaccia naturale, deve essere programmato contestualizzandolo in base all'ambiente in cui deve operare al fine di rendere il più efficiente possibile la trasformazione dei dati in informazioni utili, dall'utilizzo in contesto videoludico al contesto *automotive*. Ad oggi, quasi vent'anni dopo la nascita di questa disciplina, la ricerca, accademica e non, persiste attivamente [32].

Il settore che maggiormente ha permesso lo sviluppo di tecnologie basate su *Gesture Recognition*, e che quindi ha favorito il suo studio, è quello videoludico. Diverse piattaforme di gioco e relative periferiche di controllo, come la *Nintendo Wii* con il proprio *Wii Remote*, o ancora il *Microsoft Kinect*, che si avvale di una tecnologia basata su telecamere di natura diversa (RGB, infrarossi ed uno scanner 3D a luce strutturata), hanno contribuito alla diffusione di questa tipologia di *NUI* e ne hanno di conseguenza incrementato l'interesse mediatico, commerciale e della ricerca.

Sulla tipologia di tecnologia impiegata da quest'ultima gamma di dispositivi (telecamere), si vuole dunque concentrare questo approfondimento sull'ambito della *Gesture Recognition*.

2.1 Vision Based Gesture Recognition

Fin dalla nascita della *Gesture Recognition*, il metodo più utilizzato e più sperimentato in ambito di ricerca è quello basato su sistemi di visione, sistemi che permettono l'analisi di immagini e/o video [33]. Il funzionamento si fonda in genere sull'utilizzo di telecamere che catturano la specifica parte del corpo che esegue il movimento, nel nostro caso di studio la mano destra, al fine di tracciarne il movimento per poi essere classificato come una determinata *gesture*.

Al giorno d'oggi esistono diverse tipologie di telecamere che possono essere utilizzate in tale ambito: telecamere RGB, infrarossi e sensori per il calcolo delle *depth maps* o mappe di profondità. A tal proposito, nell'ambito della visione esistono due tipi di modelli che vengono implementati ed utilizzati per il riconoscimento di gesti: il modello tridimensionale (3D) e quello bidimensionale.

Il modello 3D presenta diverse tecniche che vengono adoperate per la rappresentazione del gesto. Tra le più note ed utilizzate vi è la tecnica definita *textured volumetric*, rappresentazione contenente dati notevolmente dettagliati non solo dello scheletro della parte interessata ma anche di altre *feature* come la pelle; il modello geometrico, meno dettagliato ma contenente più dati dello scheletro ed infine il poco utilizzato modello scheletrico [19].

Il modello bidimensionale invece, più classico, si suddivide in *modelli 2D statici*, basati su colore, forma e sagoma, e *modelli basati sul movimento (motion)* [20].

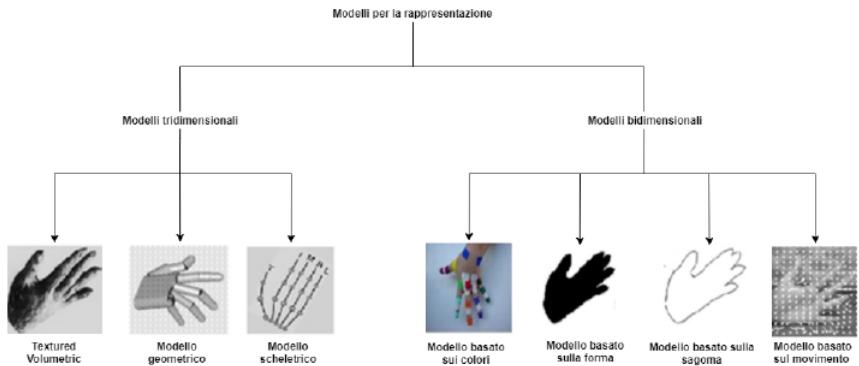


Figura 2.1: Rappresentazione grafica dei vari modelli

Un'ulteriore differenziazione nel campo del riconoscimento dei gesti, è quella tra *gesture* statiche e dinamiche. [29]

La principale differenza tra le due tipologie di gesti, come facilmente si intuisce dalla loro denominazione, consiste nel movimento compiuto o meno per attuare il determinato gesto.

Le *gesture* statiche dunque non comportano un movimento di preparazione e

termine, il gesto è presentato alla telecamera nella sua definitiva configurazione. Quelle dinamiche invece, comportano una fase di movimento che è rilevante al fine della classificazione. Ad esempio, il gesto di un pugno può essere considerato statico se la telecamera ottiene un'immagine della mano già chiusa, mentre viene considerato dinamico se quello che viene catturato dal sensore è l'intero movimento di chiusura/apertura.

Il riconoscimento di gesti affrontato in questo elaborato in particolare si concentra su quest'ultima categoria.

In letteratura sono presenti numerosi *dataset* pubblici basati su sistemi di visione, sui quali sono state applicate diverse possibili soluzioni di *Gesture Recognition*. Il loro studio ha permesso di conoscere lo stato dell'arte, individuare le tecnologie prevalentemente utilizzate (sensori RGB etc.) e identificarne i limiti se applicati nel nostro caso di studio.

2.2 Dynamic Hand Gestures Recognition: stato dell'arte

Poiché il focus dell'elaborato in questione è sul riconoscimento di gesti dinamici basato su sistemi di visione e tecniche di *deep learning*, le quali necessitano per loro natura di una grande quantità di dati, quella che viene presentata in questa sezione è una panoramica dei *dataset dinamici* disponibili in letteratura e delle relative soluzioni implementate utilizzando questi *dataset*.

Va precisato inoltre che nella ricerca e nello studio effettuato è stato preso come riferimento di applicazione il settore *automotive*, ambito generale della tesi.

2.2.1 VIVA Hand Gesture Dataset

Il *VIVA Hand Gesture Dataset* [17], è un *dataset* rilasciato in occasione dell'omonima *challenge* organizzata dal *LISA* (Laboratory for Intelligent & Safe Automobiles). È stato realizzato per studiare le attività umane ordinarie sotto contesti confusi e difficoltosi, con illuminazione variabile e occlusioni frequenti. Il *dataset* è stato acquisito mediante l'utilizzo di un dispositivo *Kinect*, di cui si è già dato un accenno, simulando situazioni di guida reali. Le tipologie di immagini raccolte dal sensore sono dunque del tipo RGB e di profondità (le diverse tipologie di immagini verranno descritte in dettaglio nel capitolo 3). I sensori per l'acquisizione del *dataset* sono stati posizionati posteriormente al guidatore, inquadrando dunque da dietro i gesti compiuti e con le telecamere ad altezza viso.

Nel *dataset* sono state identificate 19 classi di *gesture*, riprese da 8 soggetti.

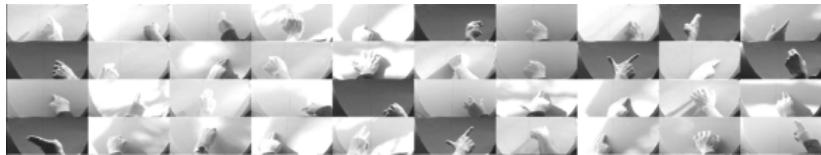


Figura 2.2: Esempio di dati provenienti dal VIVA dataset [17]

La migliore implementazione di *Gesture Recognition* per la *challenge* è quella proposta da Molchanov et al. in [28]. La soluzione consiste in un algoritmo basato su reti neurali convolutive 3D. Essa combina informazioni da scale spaziali multiple per ottenere la predizione finale. Impiega *spatio-temporal data augmentation* per avere un *training* delle reti migliore ottenendo così una percentuale di *accuracy* del 77.5%.

2.2.2 NVIDIA Dynamic Hand Gesture Dataset

Molchanov et al. in [22], per validare un metodo *online* per il riconoscimento di gesti, introducono un nuovo *dataset* dinamico multimodale, acquisito tramite sensori di profondità, a colori e infrarossi-stereo. La soluzione implementata, basata su reti neurali convolutive 3D ricorrenti che eseguono il riconoscimento sulle diverse tipologie di dati, raggiunge un valore *accuracy* del 83.8% sul *dataset* proprietario, avvicinandosi all' *accuracy* umana del 88.4% .

Il *dataset* proposto presenta 25 tipi di gesti registrati da più sensori e da differenti punti di vista. L'acquisizione è stata effettuata in un' ambiente *indoor* simulando l'interno di un veicolo. Alla registrazione del gesto hanno partecipato 20 persone, alcune delle quali hanno contribuito solo parzialmente non effettuando l'intera sessione di registrazione, raccogliendo dunque, un totale di 1532 gesti dinamici.



Figura 2.3: Ambiente di acquisizione dell'NVIDIA *dataset*; dispositivi utilizzati ((A) RGB e depth camera, (B) camera per immagini IR-stereo; modalità di acquisizione (RGB, optical flow, profondità, IR-sinistra, e disparità)).

2.2.3 Leap Motion Dynamic Hand Gesture (LMDHG)

Quest'ultimo *dataset* [18] si discosta da quelli presentati in 2.2.1 e 2.2.2 per modalità e tecnologie di acquisizione, ovvero per il contesto, che non è più quello *automotive*. Tuttavia è proprio per la presenza di un particolare sensore, il *leap motion*, che tale *dataset* è stato preso in considerazione.

Il LMDHG *dataset* contiene sequenze non segmentate temporalmente di gesti eseguiti con una e/o entrambe le mani. Il tipo di dato acquisito è l'elemento di interesse principale. Il *leap motion*, sensore di cui approfondiremo le caratteristiche nel capitolo 3, è in grado attraverso le funzionalità implementate nel SDK proprietario di estrarre le coordinate 3D dei giunti della mano, fornendo dunque un ulteriore tipologia di *feature* da poter analizzare e processare.

Il *dataset* acquisito è composto da 50 sequenze eseguite da 21 partecipanti. Ogni sequenza contiene 13 tipi di *gesture* eseguite senza ordine, più una aggiuntiva che identifica la *no-gesture*. Ogni *frame* acquisito infine, contiene le coordinate 3D di 46 giunti (23 per ogni mano).

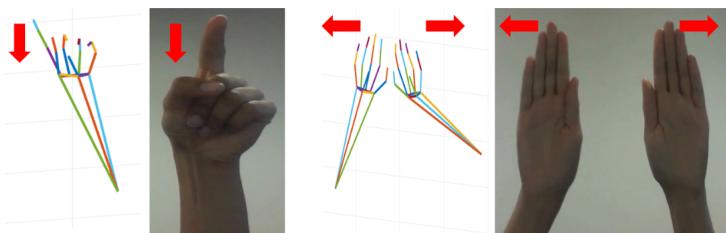


Figura 2.4: Tipologie di acquisizione - LMDHG

La presenza dunque di informazioni 3D nel *dataset*, pur non rientrando quest'ultimo nel contesto dell'elaborato, ovvero quello *automotive*, risulta essere di particolare interesse.

2.3 Considerazioni finali

I *dataset* di cui si è trattato, rappresentano lo stato dell'arte nell'ambito della *Gesture Recognition* ed offrono diverse tipologie di ambienti e modalità di acquisizione. Tuttavia, a causa di diverse scelte implementative e di vincoli imposti dal settore automobilistico, come ad esempio la struttura stessa di un abitacolo, si è resa necessaria l'acquisizione di un nuovo *dataset* che incorpori le più adatte tecnologie e modalità già analizzate e superi i limiti individuati.

Capitolo 3

Acquisizione dati

Il sistema di riconoscimento di gesti di cui tratta questo elaborato, è basato su sistemi di visione e su tecniche di *deep learning*.

Risulta necessario dunque, trattare più approfonditamente il concetto di *deep learning* e le relative problematiche che hanno condotto all'acquisizione di un nuovo e specifico *dataset*, argomento su cui è incentrato il seguente capitolo.

3.1 Deep Learning e limiti

La traduzione strettamente letterale di questo termine inglese è *apprendimento approfondito* [5]. Ed è proprio questo il centro del suo significato, perché il *deep learning*, sottocategoria del *Machine Learning*, si basa sulla creazione di modelli di apprendimento su più livelli, dunque modelli *profondi*.

L'idea che vi è dietro è abbastanza intuitiva. Si immagini di esporre una nozione. La si apprende e subito dopo ne si espone un'altra. Il nostro cervello raccoglie l'input della prima e la elabora insieme alla seconda, trasformandola e astraendola sempre di più.

Scientificamente, è corretto definire lazione del *deep learning* come l'apprendimento di dati che non sono forniti dall'uomo, ma sono appresi grazie all'utilizzo di algoritmi di calcolo statistico, storicamente ideati ispirandosi al funzionamento del cervello umano nell'interpretazioni delle immagini e del linguaggio.

L'apprendimento così ottenuto presenta la forma di una piramide: i concetti più significativi sono appresi a partire dai livelli più profondi.

Il *deep learning* nell'ultimo decennio ha subito un notevole sviluppo, raggiungendo risultati che prima erano ritenuti persino utopici. Tale successo è dovuto alle numerose conquiste nel campo informatico, relative soprattutto alla sfera dell'hardware, le quali hanno permesso un *boost* nell'elaborazione di dati e quin-

di nell'efficienza complessiva.

Elemento centrale per lo sviluppo di tecniche di *deep learning* dunque, è la necessità di un quantitativo sempre maggiore di dati, per permettere al calcolatore di fare *esperienza* su ciò che dovrà analizzare e processare.

Da quanto detto si evince come la disponibilità di dati per la specifica implementazione di tecniche di *deep learning* sia un limite importante a cui deve necessariamente essere posta una soluzione.

3.2 Contesto applicativo

Il sistema di riconoscimento di gesti che si intende sviluppare è stato ideato ideato e testato nel laboratorio RedVision, con uno specifico contesto applicativo in cui tale modello di HMI dovrà essere implementato e funzionante, ovvero quello *Automotive*.

Nel capitolo 2, sezione 2.2, si è voluto dare una panoramica dei *dataset* attualmente disponibili per il riconoscimento di gesti in ambiente automobilistico. Tali *dataset* presentano delle configurazioni hardware e di interazione, come il posizionamento dei sensori e i tipi di gesti effettuati, ben specifiche, che tuttavia si discostano dalle soluzioni ingegneristiche proposte.

È risultato necessario dunque, per seguire i vincoli imposti in fase di *design*, procedere con l'acquisizione di un nuovo *dataset* che risponda adeguatamente alle condizioni imposte.

3.3 Progettazione hardware

Come già accennato nella sezione precedente, la vera e propria acquisizione del *dataset* è stata preceduta da un'accurata ricerca della tecnologia video da utilizzare, e da uno studio della fattibilità ed efficienza del posizionamento di tali sensori all'interno dell'abitacolo.

3.3.1 Configurazione in veicolo

La prima fase di progettazione del sistema è stata quella di identificare la migliore posizione per i sensori all'interno della cabina di guida, al fine di favorire una più efficiente interazione con l'*infotainment* di bordo.

Le possibili soluzioni sperimentate per il posizionamento dei sensori sono ricadute in due principali sezioni del veicolo: la plancia, dove generalmente risiedono

i comandi per l'interazione con il computer di bordo (a), ed il tunnel centrale (b).



Figura 3.1: Possibili posizionamenti dei sensori in abitacolo: (a), (b)

Come si evince della figura 3.1, andare a disporre i sensori nella posizione (a), ovvero quella dedita al sistema di *infoteiment*, è risultata essere la soluzione da scartare a causa dell'ingombrante presenza dello strumento di interazione. La scelta in fase di progettazione pertanto è ricaduta sulla sezione (b), il tunnel centrale, per tre motivi principali:

- l'assenza di ingombri fisici che possano ostacolare il collocamento dei sensori;
- la naturale area di azione della mano che coincide con la posizione del tunnel centrale che risulta essere quindi ottimale per il rilevamento dei gesti;
- la garanzia di una minore occlusione visiva prodotta dal corpo del guidatore e dai suoi movimenti.

Inoltre, come si è potuto constatare avendo effettuato una panoramica dei *dataset* disponibili per il riconoscimento di gesti in 2.2, la posizione proposta, che differisce da quelle già analizzate per il punto di vista, apporta un elemento di novità nel settore *automotive*.

3.3.2 Sensori

Una volta individuata la posizione per il collocamento in veicolo dei sensori, si è proceduto con la ricerca e lo studio delle tecnologie video e relativi sensori attualmente disponibili nel mercato che fossero idonei in termini di ingombro fisico, alimentazione elettrica necessaria ed infine efficienza per l'implementazione di algoritmi di visione.

Inoltre, sono stati individuati una serie di requisiti al fine di implementare correttamente un'interfaccia utente naturale (NUI) nel contesto *automotive* basandosi su algoritmi di visione artificiale:

- **Invarianza alla luminosità:** i sistemi di visione devono essere affidabili anche in presenza di cambiamenti di luminosità rilevanti (per esempio durante la notte o in presenza di cattive condizioni meteo);
- **Non invasività:** i movimenti del guidatore ed la sua attenzione visiva non devono essere ostacolati durante l'attività di guida. Tale requisito risulta essere rispettato grazie al collocamento dei sensori scelto in fase di progettazione 3.3.1;
- **Performance Real Time:** il sistema di interazione deve essere veloce nel riconoscere i gesti per fornire un *feedback* efficiente del sistema.

Per rispettare questi requisiti lo studio effettuato ha portato all'individuazione di tre tipologie di immagini da acquisire e relativi sensori: immagini di profondità o *depth maps*, infrarossi (IR) ed infine RGB, queste ultime per completezza dei dati acquisiti. Inoltre, è stato acquisito un altro tipo di dato quali le coordinate 3D dei giunti della mano mediante l'uso di un particolare sensore capace di riconoscere la mano e identificarne i movimenti.

Depth Map

Le *depth map* o immagini di profondità, sono particolari tipi di immagini nelle quali ogni pixel non rappresenta l'intensità di una certa tonalità di colore, come nel caso di immagini RGB (vedi paragrafo sui sensori RGB), bensì la distanza di un determinato punto dal sensore.

Le immagini così acquisite sono formate da un solo canale (e non tre come per le immagini RGB) contenente il valore della distanza tra il dispositivo di acquisizione ed il punto all'interno della scena catturata.

In commercio esistono diversi tipi di dispositivi di acquisizione in grado di produrre questa tipologia di immagine. I più comuni sono suddivisibili in due principali categorie: gli scanner 3D a luce strutturata e i sensori *Time-of-Flight* (ToF).

Scanner 3D a luce strutturata L’idea alla base del funzionamento di questa tipologia di sensori consiste nello sfruttare le deformazioni visibili su un oggetto tridimensionale se sottoposto a *pattern* noti a priori. Per realizzare questo metodo tipicamente sono necessari due dispositivi, un proiettore che produca i *pattern* ed un sensore che acquisisca l’immagine da un’angolazione diversa rispetto a quella del proiettore.

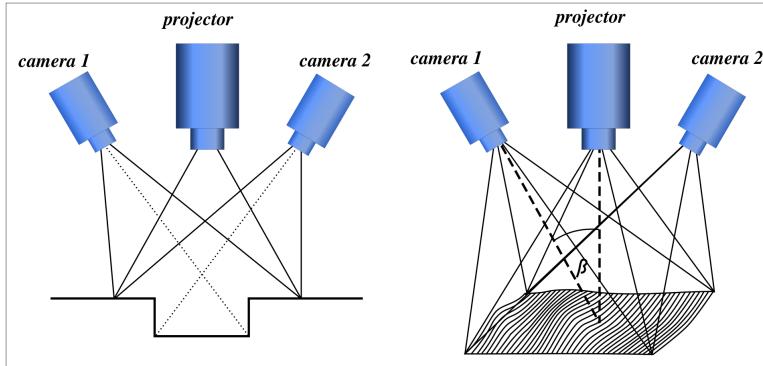


Figura 3.2: Tipica struttura per scanner 3D a luce strutturata

Tuttavia questa tipologia di sensore presenta degli svantaggi che le impediscono di soddisfare i requisiti precedentemente individuati in 3.3.2, ovvero:

- i soggetti devono stare fermi durante la proiezione del *pattern*;
- necessità di proiezione e ripresa in sequenza fissa e dunque *frame rate* più basso;
- molto dipendente ai cambiamenti di luminosità;
- sensore fisico relativamente ingombrante.

Time-of-Flight La scelta del dispositivo di acquisizione ricade quindi nella categoria dei sensori ToF. I sensori *Time-of-Flight*, a differenza degli scanner 3D a luce strutturata misurano la distanza tra oggetto e centro ottico del sensore, sfruttando il tempo percorso (da qui il termine *time of flight*) da un fascio di luce nell’incontrare l’oggetto ripreso ed essere riflesso nella direzione opposta. Il seguente meccanismo comporta due principali vantaggi che ben si adattano con i requisiti richiesti: il sensore è meno ingombrante dal punto di vista fisico e, come riportato in [31], presenta un *frame rate* di acquisizione più elevato rispetto agli scanner 3D a luce strutturata, riducendo il numero di artefatti visivi (come buchi e/o valori mancanti).

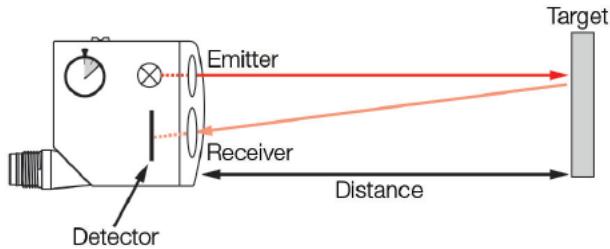


Figura 3.3: Funzionamento di un classico sensore ToF, lo strumento è composto da due parti, una che si occupa di proiettare il fascio di luce e l'altra di ricevere il fascio riflesso.

Il sensore ToF individuato per l'acquisizione del *dataset* è il dispositivo *Pico Flexx* [14], il quale presenta numerosi vantaggi:

- **Precisione ToF:** grazie a questa tecnologia il sensore è in grado di acquisire *depth map* a 16 bit con una risoluzione spaziale di 224×171 pixel con un *frame rate* che soddisfa i requisiti *real time*;
- **Fattore di forma:** il sensore presenta delle dimensioni limitate ($68mm \times 17mm \times 7.35mm$) e peso (8g). Inoltre è alimentato da una comune USB 2.0. Per queste caratteristiche risulta essere adatto ad un'applicazione nel contesto *automotive*;
- **Frame rate adattabile:** il sensore presenta diverse modalità di acquisizione: selezionando un range di acquisizione limitato, è in grado di acquisire fino a 45 immagini al secondo (45 fps);
- **Range di acquisizione:** il sensore fornisce due possibili risoluzioni *depth*, la prima copre un range di $0.5m - 4m$, mentre la seconda copre $0.1m - 1m$. Nel nostro caso, si è selezionato la seconda modalità, in questo modo, il sensore è in grado di acquisire correttamente i gesti eseguiti in prossimità del sensore. Difatti si è ipotizzato che una distanza maggiore ad un metro possa risultare inutile all'interno dell'abitacolo.



Figura 3.4: Sensore Pico FLeXX

Immagini Infrarossi

Con infrarossi ci si riferisce a telecamere in grado di acquisire la luce non visibile nella lunghezza d'onda dell'infrarosso ($0.75\mu\text{m}$ - $15\mu\text{m}$). Alcune telecamere usano illuminatori infrarossi per avere in fase di acquisizione una scena più chiara. Altre camere invece usano i più comuni sensori RGB con un filtro passa-basso infrarossi.

La luce infrarossa, da un punto di vista fisico [37], viene suddivisa in 4 bande differenti che producono informazioni (immagini) differenti:

- **Vicino (NIR)**: simili ad immagini in scala di grigi, basate sul riflesso della luce;
 - **Onda corta (SWIR)**: immagini intermedie, con caratteristiche sia di immagini NIR che termiche, basate principalmente sulla luce riflessa;
 - **Onda media (MWIR)**: immagini termiche basate su luce emessa;
 - **Onda lunga (LWIR)**: immagini termiche, basate sulla luce emessa.
- Questa è la frequenza che solitamente usano le termocamere.

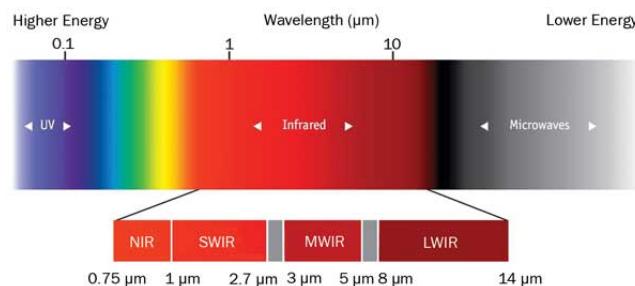


Figura 3.5: Spettro infrarosso



Figura 3.6: Esempio di immagini acquisite in bande differenti dell'IR

La scelta della tecnologia infrarossi da utilizzare è ricaduta sulla banda NIR per due motivi principali: la presenza di un sensore NIR già incorporato nel dispositivo Pico Flexx 3.3.2, utilizzato per ottenere informazioni di profondità, e per l'elevata invarianza alla luce che questa tecnologia presenta.

Immagini RGB

La struttura RGB rappresenta il metodo più comunemente utilizzato per la rappresentazioni di immagini a colori nel settore informatico. Le immagini in formato RGB sono descritte da 3 canali colore distinti, Rosso, Verde e Blu (*red, green, blue - RGB*). Ogni punto visibile, definito pixel, viene memorizzato con 3 valori a 8 bit, un valore per ogni canale colore. Il contenuto informativo è quindi espresso come un numero nell'intervallo [0 - 255] rappresentante il livello di intensità. Le immagini acquisite dunque, risultano essere equivalenti a ciò che l'occhio umano vede.

I sensori RGB presentano numerosi vantaggi: sono ampiamente diffusi e dunque economici e vi sono numerosi algoritmi pubblici che operano sui dati acquisiti da questi sensori. Tuttavia gli algoritmi non sono invarianti alla luce, non sono robusti alle occlusioni e risultano essere potenzialmente problematici con alcune etnie per il colore della pelle.

Per completezza, si è dunque aggiunto al set di dispositivi per l'acquisizione il sensore RGB LI-OV5640-USB-72 della Leopard Imaging Inc. [16], in grado di acquisire immagini con risoluzione 640×480 pixel ed avente dimensioni fisiche limitate ($40mm \times 30mm$), adatte per il contesto di utilizzo.



Figura 3.7: Sensore RGB.

In modo da rispettare il realismo dell'abitacolo di un veicolo non è stata aggiunta alcuna illuminazione esterna di supporto. Ne consegue che le immagini così acquisite presentano colori scuri e con poco contrasto.



Figura 3.8: Immagini acquisite nel *dataset*: infrarossi, RGB, *depth maps*.

Giunti tridimensionali

Infine, oltre ai più classici sensori di acquisizione di cui si è trattato nei paragrafi precedenti, si è voluto aggiungere un elemento innovativo al *dataset*, introducendo un sensore in grado di individuare la mano del guidatore e stimarne in tempo reale i giunti 3D, utilizzando le informazioni fornite dalle funzioni implementate nel'SDK proprietario: il *Leap Motion* [11].

Il sensore, grazie alle informazioni note a priori della struttura della mano insite negli algoritmi della libreria proprietaria, riconosce e tiene traccia di mani e dita operando in estrema vicinanza con le mani del soggetto, offrendo così un'elevata precisione e frequenza di tracciamento. Il Leap Motion utilizza sensori ottici e luce infrarossi, i quali sono diretti lungo l'asse *y*, ovvero verso l'alto quando il sensore si trova nella sua posizione operativa standard. Infatti il sensore utilizza un sistema di coordinate cartesiane destrose. L'origine è centrata sulla parte superiore del dispositivo. Gli assi *x* e *z* si trovano sul piano orizzontale, con l'asse *x* parallelo al bordo lungo del dispositivo. L'asse *y* è verticale, con valori positivi che aumentano verso l'alto (in contrasto con l'orientamento verso il basso della maggior parte dei sistemi di coordinate della computer grafica). L'asse *z* ha valori che incrementano positivamente verso l'utilizzatore. Il sensore è in grado di fornire un campo visivo di circa 150° e la sua operatività effettiva si estende da circa 25mm a 600mm sopra il dispositivo. Inoltre, le sue dimensioni contenute ($80\text{mm} \times 30\text{mm} \times 11.25\text{mm}$) fanno sì che venga rispettato il requisito di poco ingombro.

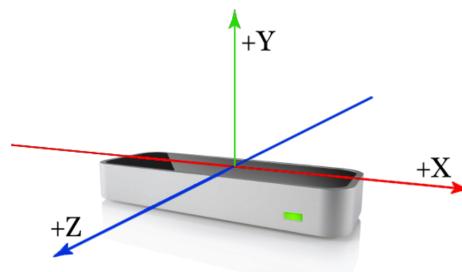


Figura 3.9: Leap motion e sistema di riferimento.

I dati acquisiti dal sensore sono di due tipologie:

- **coordinate 3D dei giunti:** tramite l' SDK proprietario il sensore è in grado di misurare le coordinate tridimensionale dei giunti della mano e in particolare delle dita, palmo e polso. Inoltre rileva velocità di movimento delle dita e i relativi angoli di imbardata, beccheggio e rollio;

- **immagini infrarossi:** il Leap Motion è dotato di due telecamere stero infrarossi attraverso le quali riesce a tenere traccia dei movimenti della mano. Le immagini fornite sono di quattro tipi: immagini distorte e rettificate acquisite dalla telecamera destra e quelle distorte e rettificate acquisite dalla telecamera sinistra.

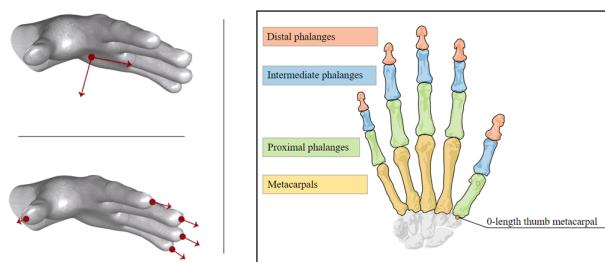


Figura 3.10: Giunti mano: coordinate palmo, punta delle dita, ossa.



Figura 3.11: Immagini acquisite dal Leap Motion: immagine telecamera destra distorta e rettificata, immagine telecamera sinistra distorta e rettificata.

3.3.3 Setup di acquisizione

Una volta individuati i sensori da utilizzare e le tipologie di dati da acquisire, si è proceduto con il collocamento dei sensori nella plancia presente nel laboratorio RedVision Lab donata da Ferrari SpA. In questo modo si è potuto ottenere una configurazione realistica della possibile collocazione finale dei sensori. I dispositivi di acquisizione dunque, come è possibile notare in figura 3.12, sono stati collocati in prossimità del tunnel centrale (qui assente) come deciso in fase di progettazione, ad una altezza dal piano di circa 35cm.



Figura 3.12: Configurazione plancia e disposizione sensori.

3.4 Gestì Dinamici

Una parte fondamentale della progettazione del *dataset* nonché di tutto il lavoro di tesi, è stata l'individuazione delle *gesture* dinamiche che il guidatore dovrà eseguire per interagire con il veicolo.

Questa fase di ideazione è stata guidata da determinati requisiti, specifici per la progettazione di un sistema di interazione il più intuitivo possibile, seguendo dunque il paradigma di NUI.

I requisiti individuati sono i seguenti:

- **facilità di esecuzione:** i gesti devono risultare facili da eseguire ed intuitivi, in modo da non compromettere la sicurezza del guidatore durante la fase di movimento del veicolo;
- **differenziazione:** ogni gesto deve poter essere facilmente distinguibile per permettere un riconoscimento automatico efficiente;
- **usabilità:** i gesti sono stati pensati per essere facilmente integrabili con i sistemi di interazione già presenti in commercio.

Una volta chiariti i requisiti necessari, la fase di ideazione ha portato all'individuazione di 12 gesti dinamici:

1. **fist (g00)**: chiusura del pugno e riapertura;
2. **pinch (g01)**: chiusura del pollice e dell'indice e riapertura;
3. **flip-over (g02)**: capovolgimento del palmo della mano e ritorno in posizione iniziale;
4. **telephone (g03)**: esecuzione del gesto della cornetta, dita chiuse ad eccezione di mignolo e pollice con leggero capovolgimento;
5. **right swipe (g04)**: *swipe* verso destra;
6. **left swipe (g05)**: *swipe* verso sinistra;
7. **top-down swipe (g06)**: *swipe* verso il basso;
8. **bottom-up swipe (g07)**: *swipe* verso l'alto capovolgendo il palmo;
9. **thumb (g08)**: chiusura della mano ad eccezione del pollice e riapertura;
10. **index (g09)**: chiusura della mano ad eccezione dell'indice e riapertura;
11. **clockwise rotation (g10)**: rotazione in senso orario della mano utilizzando solo indice e medio;
12. **councclockwise rotation (g11)**: rotazione in senso anti-orario della mano utilizzando solo indice e medio.

I gesti sono stati acquisiti suddividendo il gesto stesso in tre sotto-fasi:

- **preparazione**: il palmo viene rivolto verso i sensori prima dell'esecuzione di ogni gesto;
- **esecuzione**: viene eseguito il gesto specifico;
- **termine**: l'esecuzione del gesto viene terminata riportando la mano nella configurazione della fase di preparazione.

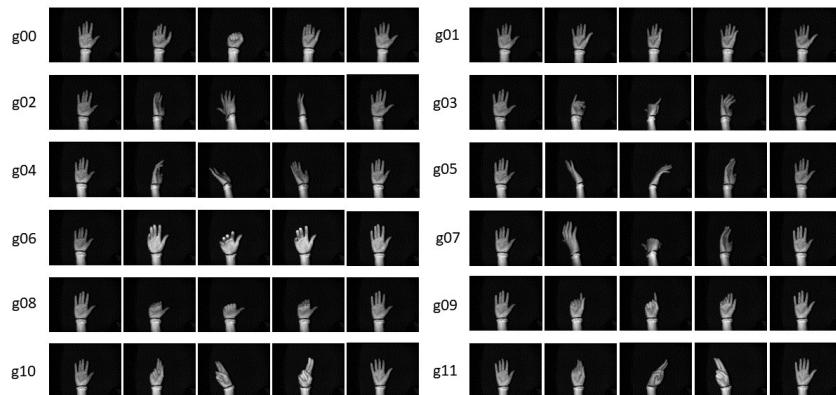


Figura 3.13: Gesti dinamici: *frame* IR di preparazione, esecuzione e termine.

3.5 Software di acquisizione

Dopo la definizione dei gesti dinamici da eseguire, si è proceduto con la scrittura del software per l’acquisizione dei dati,

Il linguaggio di programmazione utilizzato è Python, linguaggio di alto livello, che si adatta bene per l’utilizzo in ambito di visione artificiale poiché si integra facilmente e intuitivamente con librerie per l’analisi di immagini quali OpenCV e Numpy.

La realizzazione del software ha riscontrato diverse problematiche a causa della necessaria sincronizzazione tra i vari dispositivi di acquisizione utilizzati, Pico Flexx per immagini di profondità e infrarossi, il sensore LI-OV5640-USB-72 per immagini RGB, ed infine il Leap Motion per giunti 3D e immagini infrarossi. Risolte queste criticità, il processo del software di acquisizione è stato così strutturato:

1. viene generata una nuova sessione di registrazione con specifico identificativo numerico;
2. ogni sessione comprende l’esecuzione dei 12 gesti;
3. per ogni gesto vengono acquisite 3 registrazioni individuali;
4. per ogni gesto viene acquisito un minimo di 40 *frames*;
5. al termine di ogni singola registrazione viene data l’opportunità di ripetere l’acquisizione in caso di errore evidente da parte del soggetto che esegue il movimento o per malfunzionamento dei sensori;
6. se la registrazione ha avuto esito positivo si procede con l’acquisizione dei rimanenti gesti;

7. al termine dei 12 gesti viene acquisita una singola registrazione in cui il soggetto ripete tutte e 12 i movimenti consecutivamente.

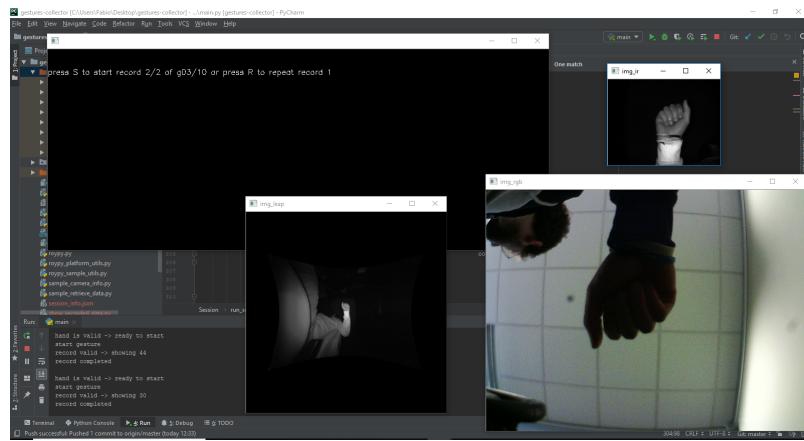


Figura 3.14: Software di acquisizione: schermata per il controllo della procedura ed esempio di immagini acquisite in tempo reale per verificare la corretta acquisizione.

3.6 Procedura di acquisizione

Determinate ed implementate tutte le componenti principali per l’acquisizione del *dataset*, si è proceduto con l’acquisizione vera e propria.

Ogni sessione di acquisizione presente nel *dataset* corrisponde alla registrazione di un singolo soggetto. Il soggetto in questione, simulando il guidatore del veicolo, è fatto accomodare sul sedile della plancia, come visibile in figura 3.15, posizionando la mano ad una distanza dai sensori di circa 43cm e rispettivamente 78cm dal piano, rimanendo dunque nel range di funzionamento del sensore Pico Flexx (3.3.2).

Viene mostrata al soggetto la posizione da tenere durante la registrazione e spiegata la procedura di acquisizione come strutturata in 3.5. Prima di eseguire ogni gesto viene mostrato una sola volta il movimento da eseguire in modo da consentire un’esecuzione del gesto diversificata da soggetto a soggetto e permettere quindi all’interno del *dateset* una certa varianza.

Il tempo di acquisizione per ogni sessione è stato di circa 6-7 minuti a seconda delle difficoltà riscontrate dal soggetto nell’esecuzione del singolo gesto.



Figura 3.15: Sessione di registrazione di un soggetto.

3.7 Conclusioni

Il *dataset* finale acquisito è composto da:

- **40 sessioni:** quaranta soggetti differenti hanno partecipato all’acquisizione del *dataset* garantendo una discreta varianza;
- **12 gesti + 1:** per ogni sessione sono stati acquisiti dodici gesti più una sequenza in cui sono stati eseguiti tutti i movimenti consecutivamente;
- **3 registrazioni:** per ogni gesto sono state effettuate tre registrazione ad esclusione della sequenza *bonus* che è stata eseguita una sola volta per sessione;
- **8 tipi di dati:** per ogni registrazione sono stati acquisiti otto tipi di dati differenti: immagini RGB, *depth map*, immagini infrarossi, coordinate 3D dei giunti della mano salvati in formato *json*, immagini telecamera sinistra distorte e rettificate e infine immagini telecamera destra distorte e rettificate;
- **40 frame:** per ogni registrazione sono stati acquisiti minimo 40 *frames*.

Il *dataset* dunque contiene complessivamente 1440 gesti per otto diversi tipi di dati. Ogni sessione così acquisita ha un peso medio di 2GB.

In fase di utilizzo nel nostro caso di studio il *dataset* è stato suddiviso in *training set* (32 sessioni) e *test set* (8 sessioni) per l’implementazione di algoritmi di *deep learning*. Il *training set* a sua volta è stato suddiviso in *training* e *validation set* (27 e 5 sessioni rispettivamente).

Capitolo 4

Metodo

Come brevemente esposto all'inizio del capitolo 3, il metodo implementato in questo lavoro di tesi per il riconoscimento di gesti si basa su algoritmi di *deep learning*, ed in particolar modo sullo sviluppo e utilizzo di reti neurali per l'analisi di immagini e dati 3D.

In questo capitolo sono esposte le diverse metodologie trattate, le reti neurali implementate, il motivo del loro utilizzo e il modo in cui sono state applicate nello specifico caso di studio.

Il framework utilizzato per l'implementazione di algoritmi di *deep learning* utilizzati nel lavoro di tesi è PyTorch nella versione 1.0.0 [15] e supporto hardware grafico Nvidia (GPU geforce 1080ti e geforce Titan X [9]).

4.1 Approccio con Reti Neurali Convulsive

Il primo approccio utilizzato per il riconoscimento di gesti è stato l'implementazione di reti neurali convulsive (*Convolutional Neural Network - CNN*) per l'analisi di immagini.

Le reti neurali convulsive sono spesso utilizzate in *task* di classificazione avendo come *input* singole immagini con uno o più canali, come immagini in scala di grigi o RGB.

Il nostro caso di studio però, essendo incentrato sul riconoscimento di gesti *dinamici*, apporta un elemento di novità, ovvero la *temporalità* del gesto. Tali reti neurali convulsive dunque, sono state implementate ed utilizzate tenendo in considerazione il fattore tempo.

Tra le varie CNN sperimentate nel lavoro di tesi, quella che ha fornito una migliore accuratezza e che di conseguenza viene riportata in questo elaborato è la *DenseNet* [25] nella sua variante *DenseNet-161*.

4.1.1 DenseNet

DenseNet è un’architettura di rete in cui ogni livello è direttamente connesso a tutti gli altri livelli in modalità *feed-forward* (all’interno di ciascun *dense block*). Per ogni livello, le *feature map* di tutti i livelli precedenti vengono trattate come input separati, mentre le proprie *feature map* vengono passate come input a tutti i livelli successivi. Questo modello di connettività offre precisioni allo stato dell’arte su CIFAR10/100 nel task di *object recognition* [27] (con o senza *data augmentation*). Sul set di dati ILSVRC 2012 (ImageNet)[30], *DenseNet* raggiunge un’accuratezza simile a quella della nota rete *ResNet* [23], ma utilizza meno della metà della quantità di parametri e richiede circa la metà del numero di FLOPs.

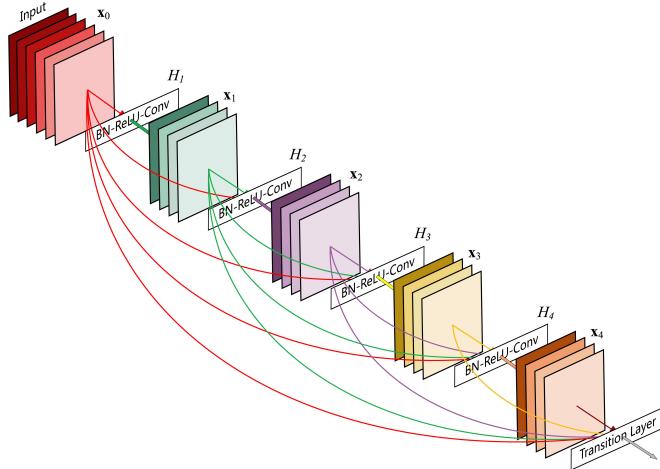


Figura 4.1: Un *Dense block* della rete con 5 *layer*.

Quest’ultimo fattore contribuisce alla motivazione per la quale è stata scelta questa rete convolutiva, ovvero il vincolo delle prestazioni real time. L’efficienza dimostrata dalla rete, fornendo prestazioni allo stato dell’arte, rientra nel requisito di affidabilità richiesto dal sistema, mentre la minore complessità computazionale rispetto alla concorrenza garantisce una maggiore garanzia prestazionale in termini di utilizzo *real time* del sistema di riconoscimento finale.

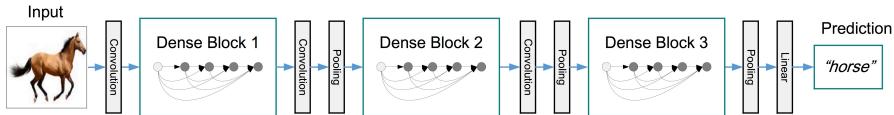


Figura 4.2: Una *deep DenseNet* con tre *dense block*.

4.1.2 Metodo proposto

Come modello di *DenseNet* è stata utilizzata la versione *DenseNet-161*.

Come espresso precedentemente (4.1), la rete è stata modificata per permettere l'elaborazione non su singole immagini ma su clip video, ovvero sequenze di n frame.

Ciascun elemento del *dataset* utilizzato è una registrazione di un gesto composta da minimo 40 *frame* per tipo di dato, come descritto in 3.5.

I frame utilizzati che vengono in seguito trasformati in tensori nella modalità spiegata successivamente sono immagini di profondità, infrarossi, RGB ed RGB convertiti in scala di grigi.

Per ogni record vengono formate clip video di durata diverse in termini di numero di frame (40, 35 e 30), impilando i singoli frame sulla terza dimensione, ovvero sul numero di canali. Ad esempio per immagini di tipo infrarossi, quindi aventi un solo canale, lo *stack* di N immagini singole della dimensione di 224×224 pixel ha prodotto un tensore di dimensioni $224 \times 224 \times N$. Mentre, per le immagini RGB, aventi 3 canali, il risultato della concatenazione è un tensore di dimensioni $224 \times 224 \times (3 \times N)$.

Come scelta dei frame da inserire nella concatenazione, al variare del numero delle immagini da utilizzare (40, 35, 30), viene considerato il frame centrale della registrazione (di lunghezza variabile a seconda della durata stessa del gesto prodotta in fase di acquisizione) e vengono utilizzati i frame nell'intervallo numerico prestabilito. Questa metodologia di selezione della sequenza è stata eseguita tenendo in considerazione che il nucleo del gesto registrato è proprio nei frame centrali acquisiti, in concordanza con la procedura di registrazione effettuata (si veda 3.4).

L'aspetto temporale del dinamismo presente nei gesti da analizzare è stato dunque realizzato impilando i frame consecutivi sulla dimensione profondità, quella dedicata ai canali dell'immagine.

Per permettere alla rete di ricevere in input un tensore tridimensionale, è stato modificato il primo *layer* convolutivo che dunque, invece di accettare come ingresso un tensore di dimensione $224 \times 224 \times 3$ (*input size* della DenseNet) ne accetta uno di dimensioni $224 \times 224 \times N$ o $224 \times 224 \times (3 \times N)$ in caso di

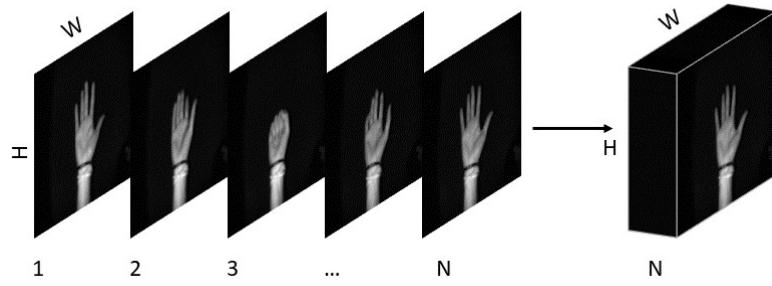


Figura 4.3: Stack di N frame. Esempio con immagini infrarossi mono canale.

immagini RGB. Inoltre è stato modificato il *layer* lineare (Fully Connected) finale per la classificazione delle 12 classi, originariamente 1000.

Nell'architettura proposta, il filtro convolutivo utilizzato è un *kernel* bidimensionale classico. Viene applicata una convoluzione 2D su un segnale di input composto da più canali. Nel caso più semplice, il valore di output di un *layer* in ingresso con dimensioni di input (N, C_{in}, H, W) e dimensioni di output $(N, C_{out}, H_{out}, W_{out})$ può essere descritta da:

$$out(N_i, C_{out_j}) = bias(C_{out_j}) + \sum_{k=0}^{C_{in}-1} weight(C_{out_j}, K) * input(N_i, K) \quad (4.1)$$

dove $*$ indica l'operatore di cross-correlazione, N è la dimensione del batch della rete in fase di *training*, C è il numero di canali del tensore di input (40, 35 o 30 nel caso di studio moltiplicato per il numero di canali originali dell'immagini, 1 o 3), H e W le dimensioni in pixel del piano di input.

4.2 Approccio con Architettura Multimodale

Il secondo metodo implementato per effettuare il riconoscimento automatico di gesti riprende le considerazioni fatte nella presentazione del metodo esposto nella sezione precedente (si ved^{<4.1>}). Difatti, la soluzione proposta consiste nella combinazione delle reti neurali convolutive già implementate e addestrate, le *DenseNet*, applicate alle diverse tipologie di dati. L'approccio utilizzato dunque, consiste nell'utilizzo di una architettura di rete multimodale.

4.2.1 Metodo proposto

La rete realizzata risulta essere composta da diversi blocchi, a seconda del tipo di combinazione di input da effettuare. Il singolo blocco è costituito da una *Densenet* già addestrata che riceverà in ingresso il tipo di dato specifico nelle modalità viste nella sezione 4.1.2.

L'*output* prodotto dal metodo è espresso dalla seguente formula:

$$\max\left(\frac{\sum_{i=0}^{N-1} \mathbf{y}_i}{N}\right) \quad (4.2)$$

dove, $\mathbf{y}_i = \{y_0, y_1, \dots, y_{M-1}\}$ è il vettore prodotto dall' i -esimo blocco dell'architettura contenente i valori predetti per le $M - 1$ classi ed N il numero di blocchi che costituiscono il modello multimodale. Il risultato dunque, è la classe che ha ottenuto il valore di predizione più alto come media tra i blocchi dell'architettura.

Figura 4.4: Architettura Multimodale.

L'architettura multimodale implementata prevede la combinazione delle seguenti tipologie di immagine:

- Depth map - IR;
- Depth map - RGB;
- Depth map - RGB gray scale;
- IR - RGB;
- IR - RGB gray scale.
- Depth map - IR - RGB;
- Depth map - IR - RGB gray scale;

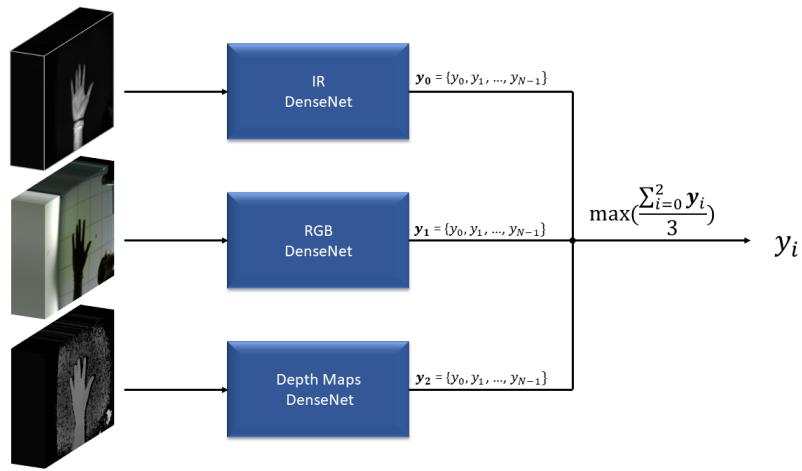


Figura 4.5: Esempio di Architettura Multimodale utilizzando immagini IR, RGB e Depth Map.

4.3 Approccio con Reti Neurali Convolutive 3D

Nelle soluzioni proposte precedentemente (4.1 e 4.2) il fattore temporale dei gesti dinamici presenti nel *dataset* è stato introdotto in un'architettura originalmente strutturata per analizzare e processare immagini singole.

Il terzo approccio che viene proposto in questo lavoro di tesi dunque, prevede l'utilizzo di reti neurali convolutive strutturate appositamente per l'analisi di sequenze temporali di immagini, quali le reti neurali convolutive 3D, e nel caso specifico di studio la rete C3D [35].

4.3.1 C3D

La soluzione proposta fornisce un approccio semplice ma efficace per l'apprendimento delle caratteristiche spaziotemporali, utilizzando reti convoluzionali tridimensionali (*3D ConvNet*) pre-allenate su un vasto *dataset* supervisionato di video [26].

I risultati raggiunti dalla rete in questione, utilizzando un semplice layer lineare per la classificazione superano metodi allo stato dell'arte in *task* come ad'esempio l'*action recognition* [1]. Infine, sono concettualmente molto semplici e facili da addestrare ed utilizzare [35].

La principale differenza che vi è tra la rete C3D e quelle proposte nelle sezioni precedenti, come già accennato, è la struttura convolutiva ideata specificatamente per gestire l'aspetto temporale.

Nell'architettura proposta in 4.1, il filtro convolutivo utilizzato è un *kernel* bidimensionale classico. Viene applicata una convoluzione 2D su un segnale di input composto da più canali. Nel caso più semplice, il valore di output di un *layer* in ingresso con dimensioni di input (N, C_{in}, H, W) e dimensioni di output $(N, C_{out}, H_{out}, W_{out})$ può essere descritta da:

$$out(N_i, C_{out_j}) = bias(C_{out_j}) + \sum_{k=0}^{C_{in}-1} weight(C_{out_j}, K) * input(N_i, K) \quad (4.3)$$

dove $*$ indica l'operatore di cross-correlazione, N è la dimensione del batch della rete in fase di *training*, C è il numero di canali del tensore di input (40, 35 o 30 nel caso di studio), H e W le dimensioni in pixel del piano di input.

Nella rete C3D a differenza della rete convolutiva già trattata (si veda 4.1), il filtro convolutivo applicato non è bidimensionale bensì tridimensionale. Dunque, la rete applica delle convoluzioni 3D sul segnale di ingresso anch'esso com-

posto da più piani. Nel caso più semplice, il valore di output di un *layer* in ingresso con dimensioni di input (N, C_{in}, D, H, W) e dimensioni di output $(N, C_{out}, D_{out}, H_{out}, W_{out})$ può essere descritto da:

$$out(N_i, C_{out_j}) = bias(C_{out_j}) + \sum_{k=0}^{C_{in}-1} weight(C_{out_j}, K) * input(N_i, K) \quad (4.4)$$

dove però $*$ indica l'operatore di cross-correlazione 3D, $C_{in,out}$ il numero di canali presenti (40, 35, 30) e D il numero di dimensioni del tensore in ingresso (3 per immagini RGB e 1 per *depth map* e IR).

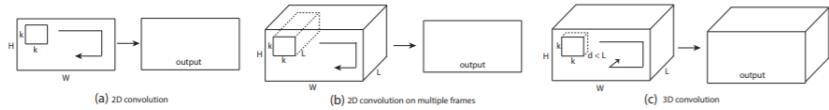


Figura 4.6: Operazioni di convoluzione 2D e 3D. a) L'applicazione della convoluzione 2D su un'immagine produce un'immagine. b) L'Applicazione della convoluzione 2D su un volume video (più fotogrammi come più canali (vedi 4.1)) genera un'immagine. c) L'applicazione della convoluzione 3D su un volume video produce un altro volume, preservando le informazioni temporali del segnale di ingresso [35].

4.3.2 Metodo proposto

La rete adoperata presenta la seguente struttura: ha 8 convoluzioni, 5 *max-pooling* e 2 livelli completamente connessi, seguiti da un *layer softmax* di output. Tutti i *kernel* di convoluzione 3D sono $3 \times 3 \times 3$ con *stride* 1 in entrambe le dimensioni spaziali e temporali. Il numero di filtri è indicato in ciascuna casella. I livelli di *pool* 3D sono indicati da *pool1* a *pool5*. Tutti i *kernel* di *pooling* sono $2 \times 2 \times 2$, ad eccezione di *pool1* che è $1 \times 2 \times 2$. Ogni *layer* completamente connesso ha 4096 unità di uscita [35]. Tuttavia sono state apportate delle modifiche per il caso specifico di studio ma che non ha prodotto grosse modifiche alla struttura originaria del modello appena esposta. ad eccezione del layer lineare terminale che è stato modificato per la classificazione di 12 classi in sostituzione delle 487 presenti nella configurazione implementata.



Figura 4.7: Architettura C3D. [35].

Il tensore fornito alla rete come input è stato implementato secondo la struttura richiesta dalla rete, ovvero avente dimensioni (N, C_{in}, D, H, W) , con $D = 3$ in caso di immagini RGB o $D = 1$ negli altri casi, e $H = W = 112$.

4.4 Approccio con Reti Neurali Ricorrenti

I metodi proposti nelle sezioni precedenti hanno come caratteristica fondante comune l'utilizzo di reti neurali convolutive. Tali reti, come già spiegato, si basano sull'analisi e l'elaborazione di dati relativi alla visione, quali le immagini di profondità, immagini infrarossi e RGB. Inoltre, l'aspetto temporale legato ai gesti dinamici è stato trattato in queste soluzioni come una terza dimensione aggiuntiva (impilando i frame sulla dimensione dedicata ai canali della singola immagine) allo spazio già identificato da altezza (H) e larghezza (W) in pixel delle immagini.

Il metodo che viene presentato in questa sezione cambia l'approccio al problema utilizzato in precedenza. Non vengono più utilizzate reti neurali convolutive, ma reti in grado di operare direttamente sulla temporalità dei gesti, utilizzando dati questa volta di natura prettamente numerica.

L'approccio proposto si basa sull'uso di reti neurali ricorrenti, nel caso specifico di *Long Short Term Memory network* (LSTM) [24], per l'analisi dei giunti forniti dall' SDK del sensore *Leap Motion* utilizzato in fase di acquisizione del *dataset*.

4.4.1 LSTM

Le reti LSTM, sono un tipo speciale di RNN (Reti Neurali Ricorrenti) in grado di apprendere le dipendenze a lungo termine. Risultano essere una soluzione valida su una grande varietà di problemi, come per il *language modelling* [34] o per il *sentiment analysis* [36] e sono ora ampiamente utilizzati.

Tali reti sono esplicitamente progettate per evitare il problema della dipendenza a lungo termine. Sono infatti realizzate per ricordare le informazioni per lunghi periodi di tempo.

Tutte le reti neurali ricorrenti hanno la forma di una catena di moduli, singole reti neurali, ripetuti. Nelle RNN standard, questo modulo ripetitivo presenta una struttura molto semplice, ad esempio un singolo layer *tanh*. Le reti LSTM hanno anch'esse la struttura a catena appena descritta, ma il modulo ripetuto ha una struttura diversa. Invece di avere un singolo layer di rete neurale, ve ne sono quattro, che interagiscono in modo da conservare l'informazione temporale acquisita, andando a rimuovere o aggiungere contenuto informativo alla cella di stato o *cell state* (elemento principale delle reti ricorrenti), in maniera regolata da strutture definite *gate* [12].

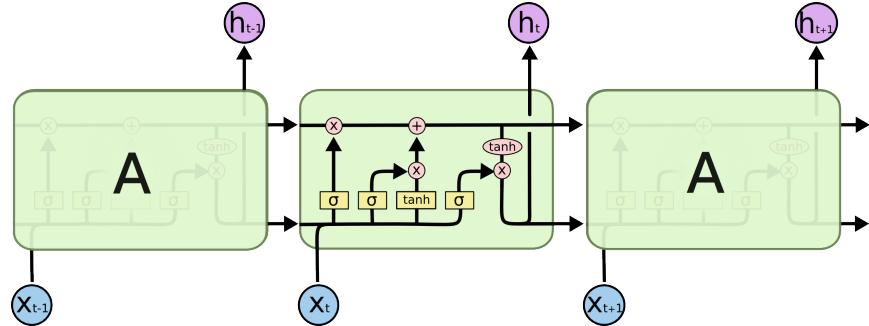


Figura 4.8: Architettura di una rete LSTM. X_t rappresenta l'input temporale, h_t l'output (hidden state) prodotto.

Per ogni elemento nella sequenza di input, ogni *layer* calcola la seguente funzione:

$$\begin{aligned} i_t &= \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{(t-1)} + b_{hi}) \\ f_t &= \sigma(W_{if}x_t + b_{if} + W_{hf}h_{(t-1)} + b_{hf}) \\ g_t &= \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{(t-1)} + b_{hg}) \\ o_t &= \sigma(W_{io}x_t + b_{io} + W_{ho}h_{(t-1)} + b_{ho}) \\ c_t &= f_t c_{(t-1)} + i_t g_t \\ h_t &= o_t \tanh(c_t) \end{aligned} \quad (4.5)$$

dove: h_t è lo *hidden state* all'istante t , c_t è la *cell state* all'istante t , x_t è l'input al momento t , $h_{(t-1)}$ è lo *hidden state* del *layer* all'istante $t-1$ o lo *hidden state* iniziale al tempo 0. i_t , f_t , g_t , o_t sono rispettivamente i *gate* di *input*, *forget*, *cell* e *output*. Infine *sigma* è la funzione sigmoide.

In una rete LSTM multi livello o *multilayer*, l'input $i_t^{(l)}$ del l -esimo *layer* ($l \geq 2$) è lo *hidden state* $h_t^{(l-1)}$ del livello precedente, moltiplicato per un fattore di *dropout* [13].

4.4.2 Metodo proposto

La rete LSTM implementata presenta la seguente struttura:

- **hidden size** = 256: ovvero il numero di *feature* nello *hidden state* h ;
- **num_layers** = 2: ovvero il numero di *layer* ricorrenti. Avere un numero di *layer* maggiore di 1 vuol dire impilare insieme due reti LSTM, con la seconda che riceve come input l'output del primo *layer* e calcola il risultato finale;
- **input size** = 145: ovvero il numero delle *feature* presenti nel tensore di input;

- **2 layer lineari:** l'output del secondo livello della rete LSTM, che nell'implementazione effettuata consiste nel tensore di output relativo all'ultimo istante temporale t (dimensioni: $(seq_len, batch, num_directions * hidden_size)$, con $num_directions = 1$) viene processato da due *layer lineari* che forniscono la classificazione finale dei gesti.

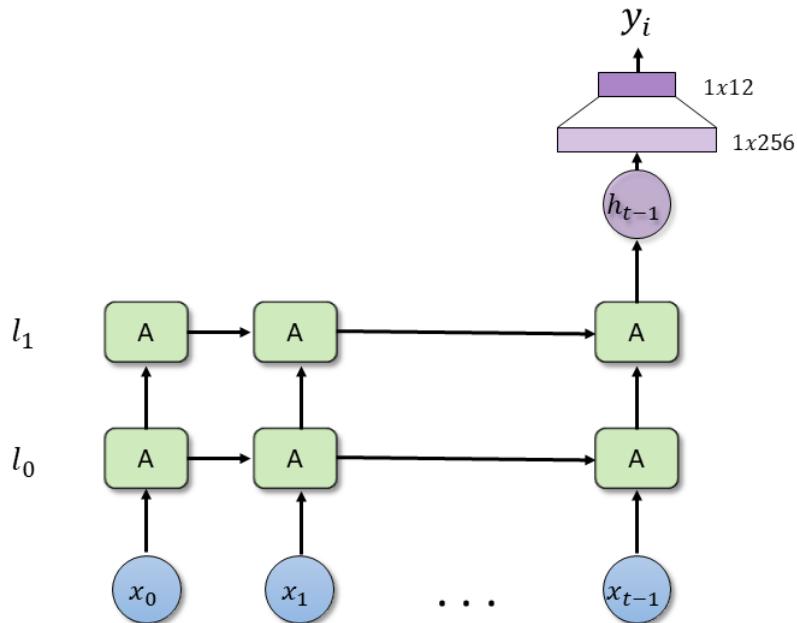


Figura 4.9: Architettura della rete LSTM implementata. h_i tensore i -esimo di input con t variabile (40, 35, 30), l_i livello della rete, A cella lstm, e y_i classe predetta.

Tensore di Input Le dimensioni del tensore di input che riceve la rete in ingresso sono le seguenti: $(seq_len, batch, input_size)$ dove seq_len indica la lunghezza temporale della sequenza che, come per le reti già esposte, può assumere valori 40, 35 o 30 (ovvero il numero di frame acquisiti per l' i -esimo gesto). $input_size$ è il parametro che determina la dimensione del tensore in termini di numero di *feature* presenti. Va ricordato che per ogni frame acquisito in fase di collezione del *dataset* sono state raccolte le coordinate 3D dei giunti della mano e altre misure per un totale di 675 valori. Di questi 675 valori ne sono stati selezionati 145, ritenute le più discriminative al fine di identificare il gesto dinamico.

Le *feature* selezionate per il tensore di input sono le seguenti:

- **posizione del palmo:** la posizione centrale del palmo in millimetri rispetto all'origine del Leap Motion. Coordinate x, y, z, imbardata, beccheggio

e rollio;

- **normale del palmo:** il vettore normale al palmo. Coordinate x, y, z, imbardata, beccheggio e rollio;
- **velocità del palmo:** il tasso di variazione della posizione del palmo in millimetri/secondo. Coordinate x, y, z, imbardata, beccheggio e rollio;
- **ampiezza palmo:** la larghezza media della mano (escluse le dita o il pollice).
- **forza nel *pinch*:** la forza di un *pinch* tra il pollice e la punta del dito più vicina come valore nell'intervallo [0..1].
- **forza nel *grab*:** la forza di un *grab* come valore nell'intervallo [0..1].
- **direzione:** la direzione dalla posizione del palmo nei confronti delle dita. Coordinate x, y, z, imbardata, beccheggio e rollio;
- **centro della sfera:** il centro di una sfera che si adatta alla curvatura della mano. Coordinate x, y, z, imbardata, beccheggio e rollio;
- **posizione del polso:** la posizione del polso della mano riconosciuta. Coordinate x, y, z, imbardata, beccheggio e rollio;
- **posizione della punta di ciascun dito:** Coordinate x, y, z, imbardata, beccheggio e rollio;
- **velocità della punta di ciascun dito:** Coordinate x, y, z, imbardata, beccheggio e rollio;
- **direzione della punta di ciascun dito:** Coordinate x, y, z, imbardata, beccheggio e rollio;
- **lunghezza di ciascun dito:** lunghezza media del dito;
- **ampiezza di ciascun dito:** ampiezza media del dito.
- **estensione di ciascun dito:** se il dito è o non è steso.

Sono stati invece scartate le *feature* definite dall'SDK del Leap Motion come *pointables*, contenenti informazioni sulla struttura e le coordinate in tempo reale delle ossa delle dita.

Capitolo 5

Risultati sperimentali

In questo capitolo sono riportati i risultati ottenuti impiegando i modelli e le architetture esposte nel capitolo precedente (capitolo4) per il riconoscimento di gesti.

Gli esperimenti di valutazione dei metodi sono stati effettuati utilizzando il *dataset* acquisito e descritto in questo lavoro di tesi (si veda 3). Il *dataset* è stato suddiviso in 3 parti: 26 delle 40 sessioni totali presenti sono state utilizzate in fase di *training* dei metodi implementati, 6 per la fase di *validation* in *training*, e le rimanenti 8 sessioni sono state utilizzate per il *test* e valutazione dei modelli di reti neurali proposti.

Va ricordato inoltre che i soggetti ripresi per l'acquisizione del *dataset* differiscono di sessione in sessione, ovvero ogni sessione di registrazione è stata effettuata interamente da un solo soggetto, garantendo conformità alla fase di valutazione.

5.1 Metriche

Per ogni metodo proposto sono riportati i risultati di accuratezza media (*accuracy*) nel riconoscimento dei 12 gesti per le varie tipologie di input utilizzati relativamente allo specifico metodo (*depth map*, immagini IR, immagini RGB, immagini RGB in scala di grigi e coordinate 3D dei giunti), e, ove effettuata, una comparazione dei risultati utilizzando dimensioni di sequenze diverse (40, 35 e 30 *frame*).

Infine, per ogni metodo verrà riportato l'*accuracy* nel riconoscimento di ciascuna delle 12 classi di gesti esclusivamente per la configurazione del modello che ha ottenuto il punteggio di *accuracy* media migliore, e le relative matrici di confusione (normalizzate e non).

I punteggi di *accuracy* sono riportati in valore decimale [0.0 ... 1.0], con 1.0 rappresentante il 100% dell'*accuracy*.

5.2 Approccio con Reti Neurali Convolutive

5.2.1 Modalità di Training

In questa sezione sono riportati il procedimento e la configurazione di *training* utilizzati per il metodo proposto.

Per ogni tipo di dato, *depth map*, immagini infrarossi, RGB ed RGB convertite in scali di grigi, è stato effettuato il training della rete per le diverse configurazioni di concatenazione: 40, 35 e 30 frame.

Per il training della rete sono stati utilizzati i pesi forniti nell'implementazione della *DenseNet-161* di PyTorch effettuando un *fine-tuning* per i layer modificati.

È stata effettuata della *data augmentation* per incrementare la precisione della rete e renderla invariante a piccole variazioni. Sono state attuate tecniche per l'incremento del numero e della varietà dei dati di training. Nello specifico sono stati applicati *random flip* e *random crop* con probabilità del 50% alla dimensione 224×224 pixel.

Inoltre, i dati prima di essere forniti alla rete come input sono stati normalizzati sottraendo la media e dividendo per la varianza, calcolate per ogni modalità di immagine su tutto il *training set*, in modo da ottenere media nulla e varianza unitaria.

La rete è stata allenata per 50 epoche con *batch size* di 8, *learning rate* pari a 10^{-2} e *momentum* di 0.5. La funzione di *loss* utilizzata è la *binary categorical cross-entropy* e lo stochastic gradient descent (SGD) come ottimizzatore.

I risultati ottenuti in fase di *training* e *validation* per le diverse modalità sono visibili nei grafici a seguire.

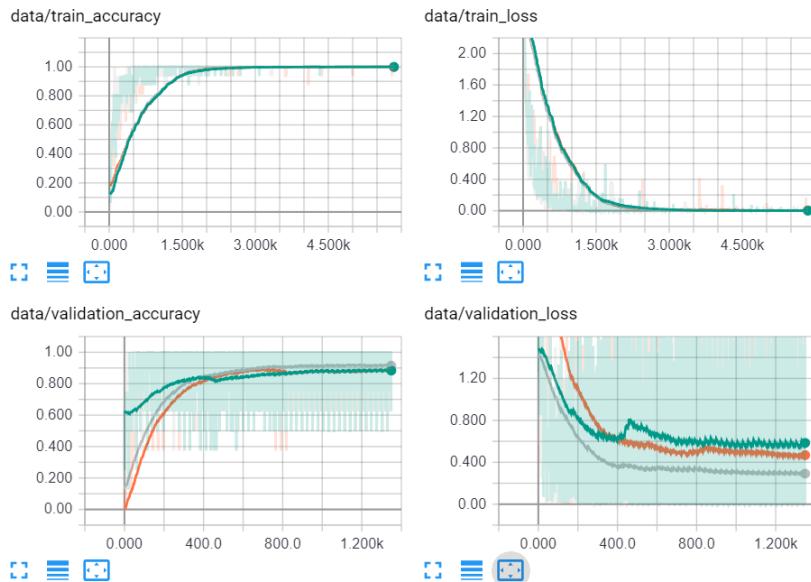


Figura 5.1: Input della rete: depth map. Grafico di accuracy e loss in *training* e *validation* per N frame diversi: in verde $N = 40$; in grigio $N = 35$; in arancio $N = 30$.

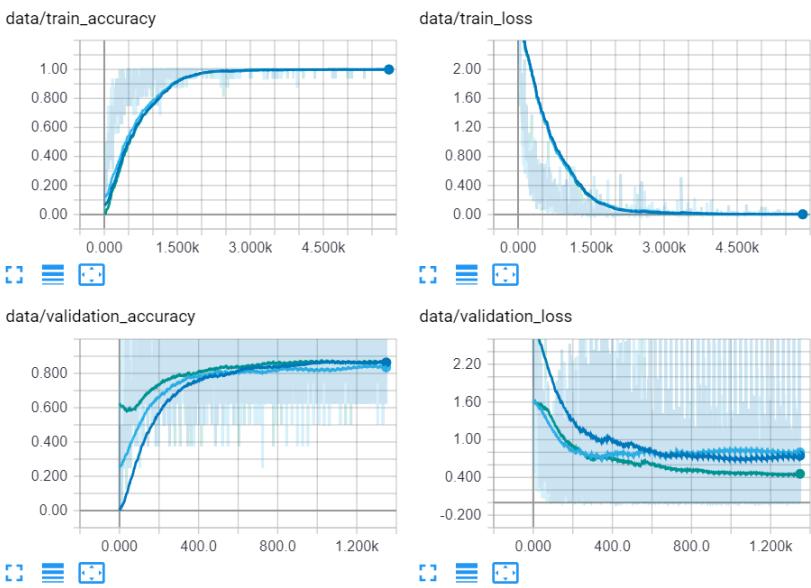


Figura 5.2: Input della rete: immagini IR. Grafico di accuracy e loss in *training* e *validation* per N frame diversi: in blu $N = 40$; in azzurro $N = 35$; in verde $N = 30$.

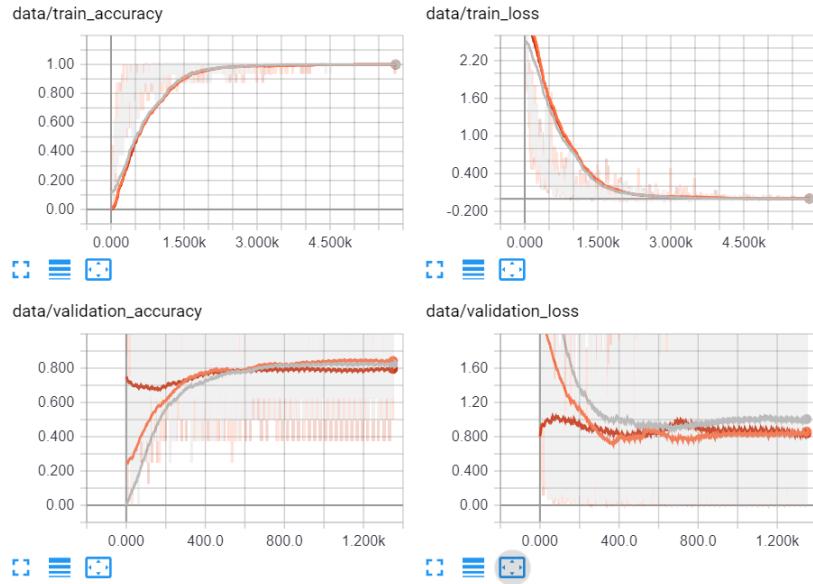


Figura 5.3: Input della rete: immagini RGB. Grafico di accuracy e loss in *training* e *validation* per N frame diversi: in grigio $\mathcal{N} = 40$; in arancio $\mathcal{N} = 35$; in rosso $\mathcal{N} = 30$.

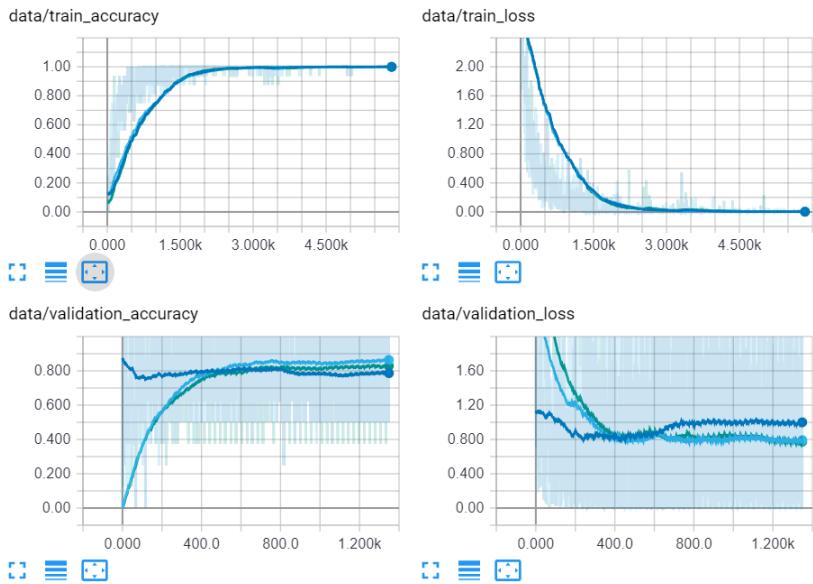


Figura 5.4: Input della rete: immagini RGB in scala di grigi. Grafico di accuracy e loss in *training* e *validation* per N frame diversi: in blu $\mathcal{N} = 40$; in azzurro $\mathcal{N} = 35$; in verde $\mathcal{N} = 30$.

5.2.2 Risultati

In questa sezione vengono presentati in maniera tabulare i risultati ottenuti nella valutazione del metodo proposto nella sezione 4.1 e le relative matrici di confusione.

La valutazione è stato effettuato processando i gesti presenti nelle 8 sessioni del *dataset* dedicate al test, riproponendo la configurazione del metodo proposto in fase di training.

Le tipologie di dati utilizzate in fase di test per le quali sono riportate le relative metriche sono:

- **Depth map** (tab. 5.2 e fig. 5.5)
- **Immgini IR** (tab. 5.3 e fig. 5.6)
- **Immagini RGB** (tab. 5.4 e fig. 5.7)
- **Immagini RGB grayscale** (tab. 5.5 e fig. 5.8)

Accuray CNN per tipo di input				
Seq. len.	Depth map	IR	RGB	RGB grayscale
40	0,903	0,861	0,837	0,871
35	0,903	0,878	0,851	0,865
30	0,861	0,892	0,837	0,851

Tabella 5.1: Accuracy del metodo CNN al variare del tipo di input utilizzato e della lunghezza in frame della sequenza. In grassetto è evidenziato il risultato migliore.

Successivamente sono riportate le tabelle di accuracy per classe [g0 ... g11] delle configurazioni di input e lunghezza della sequenza riportate nella tabella 5.1 che hanno ottenuto il miglior risultato e le relative matrici di confusione. Il totale delle sequenze analizzate in fase di test è pari a 288, 24 sequenze per gesto.

Accuracy per classe - CNN - Depth Map - 40			
Class	Accuracy	Class	Accuracy
g0	0,958	g6	1,000
g1	1,000	g7	0,958
g2	0,917	g8	1,000
g3	0,917	g9	1,000
g4	0,958	g10	0,583
g5	0,875	g11	0,667

Tabella 5.2: Accuracy per classe per il tipo di input depth map e lunghezza sequenza = 40 frame.

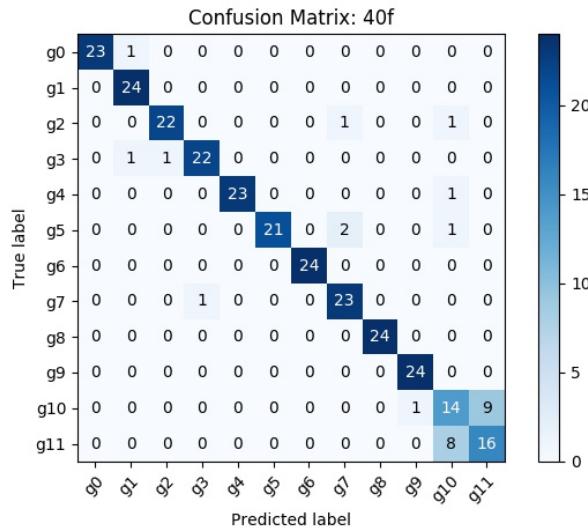


Figura 5.5: Matrice di confusione per il tipo di input: depth map. Lunghezza sequenza = 40 frame

Accuracy per classe - CNN - IR - 30			
Class	Accuracy	Class	Accuracy
g0	0,958	g6	1,000
g1	1,000	g7	0,917
g2	1,000	g8	1,000
g3	0,875	g9	1,000
g4	0,875	g10	0,750
g5	0,875	g11	0,458

Tabella 5.3: Accuracy per classe per il tipo di input immagini IR e lunghezza sequenza = 30 frame.

Accuracy per classe - CNN - RGB - 35			
Class	Accuracy	Class	Accuracy
g0	0,792	g6	1,000
g1	1,000	g7	0,792
g2	0,958	g8	1,000
g3	0,917	g9	0,958
g4	0,875	g10	0,542
g5	0,792	g11	0,583

Tabella 5.4: Accuracy per classe per il tipo di input immagini RGB e lunghezza sequenza = 35 frame.

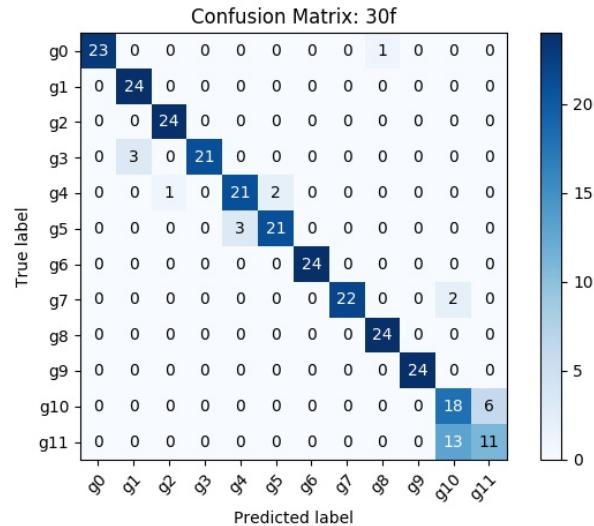


Figura 5.6: Matrice di confusione per il tipo di input: immagini IR. Lunghezza sequenza = 30 frame

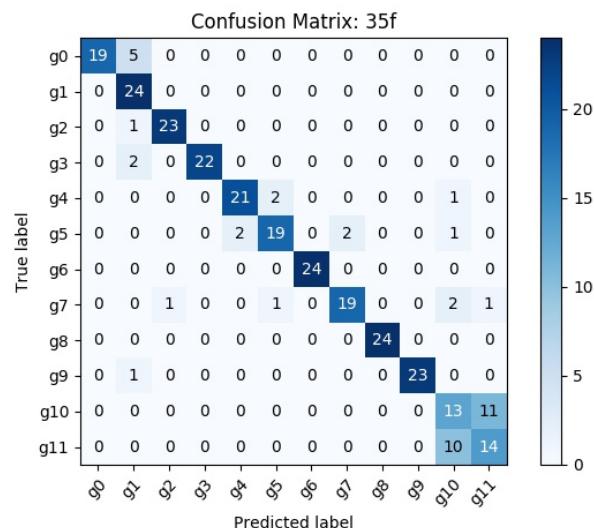


Figura 5.7: Matrice di confusione per il tipo di input: immagini RGB. Lunghezza sequenza = 35 frame

Accuracy per classe - CNN - RGB gs - 40			
Class	Accuracy	Class	Accuracy
g0	0,958	g6	1,000
g1	1,000	g7	0,875
g2	0,875	g8	1,000
g3	0,958	g9	1,000
g4	0,875	g10	0,500
g5	0,792	g11	0,625

Tabella 5.5: Accuracy per classe per il tipo di input immagini RGB grayscale e lunghezza sequenza = 40 frame.

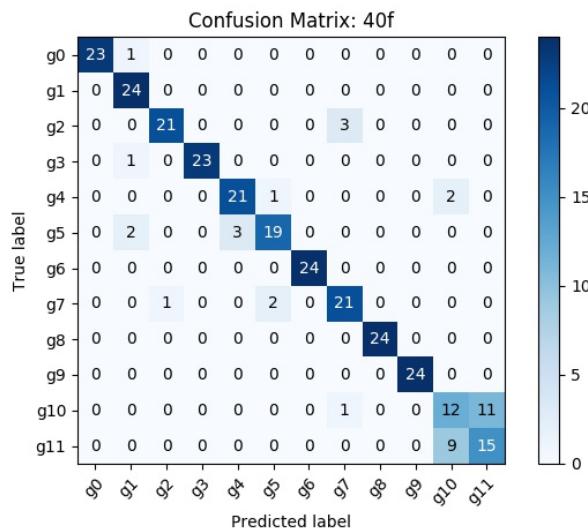


Figura 5.8: Matrice di confusione per il tipo di input: immagini RGB grayscale. Lunghezza sequenza = 40 frame

5.2.3 Analisi Risultati

Dalle metriche riportate (si veda tab. 5.1) si può notare come il metodo sia più accurato processando input del tipo *depth map* e con sequenze di lunghezza 40 frame. Tale risultato è dovuto al maggiore contenuto informativo insito nelle immagini di profondità. Difatti i gesti proposti essendo dinamici coinvolgono maggiormente le tre dimensioni spaziali aggiungendo dunque più informazione rispetto alle più statiche immagini 2D.

Inoltre si può osservare come al variare della lunghezza della sequenza in input i cambiamenti nei risultati siano minimi. Ciò è dovuto al fatto che il nucleo del gesto risulta essere sufficientemente contenuto in tutte le sequenze anche al variare della lunghezza.

È osservabile come le classi che ottengono valori di accuracy peggiori sono le classi g10 e g11, rappresentanti i due gesti dinamici più lunghi presenti nel *dataset*. Questo fenomeno è notabile attraverso la relativa matrice di confusione riportata (fig. 5.5) e dalla tabella contenente l'accuracy per classe del metodo (tab. 5.2). È possibile notare come l'accuracy cali drasticamente per le classi g10 e g11 che, come visibile nella matrice di confusione, vengono spesso confuse. Ciò è dovuto alla notevole similarità del gesto compiuto che consiste in una rotazione dell'indice e del dito medio con pugno chiuso in senso orario per la classe g10 e in senso opposto per la classe g11. Tale movimento può essere confuso dall'occhio umano e ne consegue che anche il sistema per queste due classi si trovi in difficoltà.

5.3 Approccio con Architettura Multimodale

5.3.1 Risultati

In questa sezione vengono presentati i risultati ottenuti nella valutazione del metodo proposto in 4.2 e le relative matrici di confusione.

La valutazione è stato effettuato processando i gesti presenti nelle 8 sessioni del *dataset* dedicate al test.

Il presente modello è stato valutato esclusivamente con sequenza di lunghezza uguale a 40 *frame*. Tale configurazione è stata adottata poiché come visto nella sezione precedente ed in particolare nella tabella 5.1, i risultati al variare della lunghezza della sequenza non subiscono cambiamenti particolarmente rilevanti ai fini della precisione del metodo, ed inoltre perché la configurazione migliore rilevata per il metodo precedente processa sequenze di tale lunghezza.

Le tipologie di modalità di input utilizzate in fase di test per le quali sono riportate le relative metriche sono:

- **Depth map, IR** (tab. 5.7 e fig. 5.9)
- **Depth map, RGB** (tab. 5.8 e fig. 5.10)
- **Depth map, RGB grayscale** (tab. 5.9 e fig. 5.11)
- **IR, RGB** (tab. 5.10 e fig. 5.12)
- **IR, RGB grayscale** (tab. 5.11 e fig. 5.13)
- **Depth map, IR, RGB** (tab. 5.12 e fig. 5.14)
- **Depth map, IR, RGB grayscale** (tab. 5.13 e fig. 5.15)

Accuracy per modalità di input	
Input	Accuracy
Depth Map, IR	0,920
Depth Map, RGB	0,896
Depth Map, RGB Grayscale	0,910
IR, RGB	0,865
IR, RGB Grayscale	0,865
Depth Map, IR, RGB	0,910
Depth Map, IR, RGB Grayscale	0,899

Tabella 5.6: *Accuracy* media dell'Architettura Multimodale per ciascun tipo di input.

Successivamente sono riportate le tabelle di accuracy per classe [g0 ... g11] delle configurazioni di input riportate nella tabella 5.6 e le relative matrici di confusione. Il totale delle sequenze analizzate in fase di test è pari a 288, 24 sequenze per gesto.

Accuracy per classe			
Arch. Multimod. - Depth Map, IR			
Class	Accuracy	Class	Accuracy
g0	1,000	g6	1,000
g1	1,000	g7	0,958
g2	0,917	g8	1,000
g3	0,958	g9	1,000
g4	0,958	g10	0,625
g5	0,917	g11	0,708

Tabella 5.7: Accuracy per classe per il tipo di input depth map - IR.

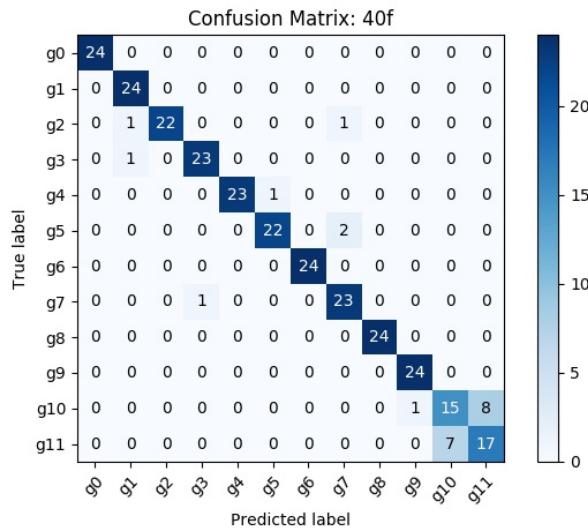


Figura 5.9: Matrice di confusione per il tipo di input: depth map - IR.

Accuracy per classe			
Arch. Multimod. - Depth Map, RGB			
Class	Accuracy	Class	Accuracy
g0	1,000	g6	1,000
g1	1,000	g7	0,958
g2	0,917	g8	1,000
g3	1,000	g9	1,000
g4	0,917	g10	0,500
g5	0,833	g11	0,625

Tabella 5.8: Accuracy per classe per il tipo di input depth map - RGB.

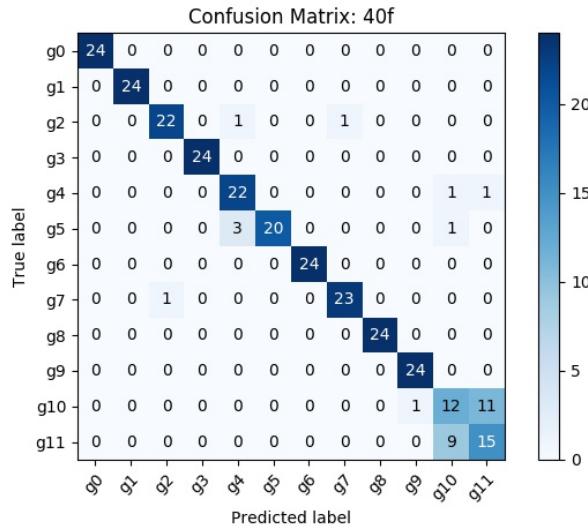


Figura 5.10: Matrice di confusione per il tipo di input: depth map - RGB.

Accuracy per classe			
Arch. Multimod. - Depth Maps, RGB gs			
Class	Accuracy	Class	Accuracy
g0	1,000	g6	1,000
g1	1,000	g7	0,917
g2	0,958	g8	1,000
g3	0,958	g9	1,000
g4	0,958	g10	0,583
g5	0,917	g11	0,625

Tabella 5.9: Accuracy per classe per il tipo di input depth map - RGB grayscale.

Accuracy per classe			
Arch. Multimod. - IR, RGB			
Class	Accuracy	Class	Accuracy
g0	0,958	g6	1,000
g1	1,000	g7	0,917
g2	0,917	g8	1,000
g3	0,875	g9	1,000
g4	0,875	g10	0,500
g5	0,917	g11	0,583

Tabella 5.10: Accuracy per classe per il tipo di input IR - RGB.

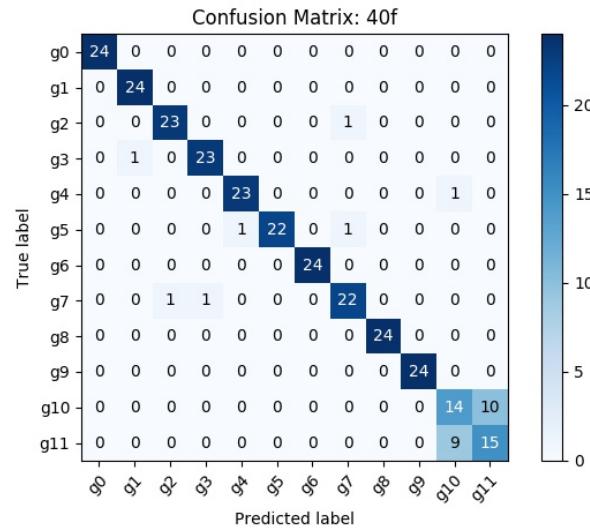


Figura 5.11: Matrice di confusione per il tipo di input: depth map - RGB grayscale.

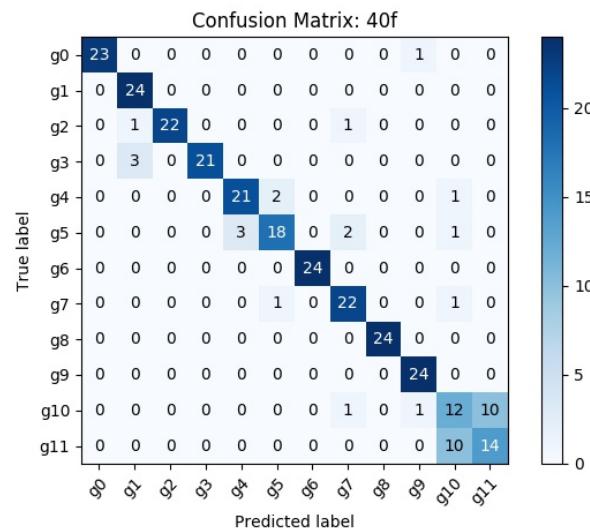


Figura 5.12: Matrice di confusione per il tipo di input: IR - RGB.

Accuracy per classe Arch. Multimod. - IR, RGB gs			
Class	Accuracy	Class	Accuracy
g0	0,958	g6	1,000
g1	1,000	g7	0,917
g2	0,833	g8	1,000
g3	0,875	g9	1,000
g4	0,875	g10	0,542
g5	0,750	g11	0,625

Tabella 5.11: Accuracy per classe per il tipo di input IR - RGB grayscale.

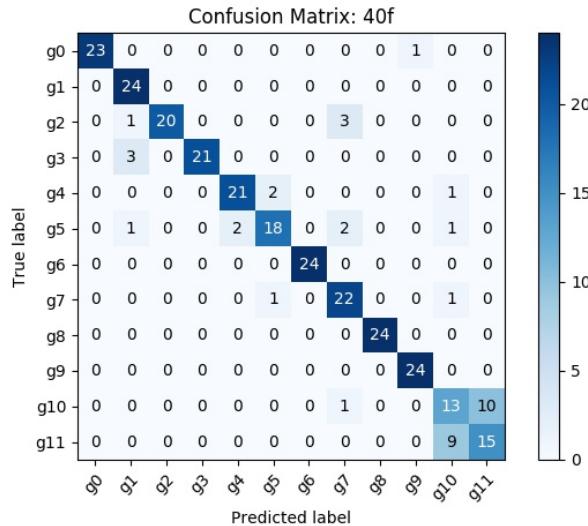


Figura 5.13: Matrice di confusione per il tipo di input: IR - RGB Grayscale.

Accuracy per classe Arch. Multimod. - Depth Map, IR, RGB			
Class	Accuracy	Class	Accuracy
g0	1,000	g6	1,000
g1	1,000	g7	0,917
g2	0,958	g8	1,000
g3	0,917	g9	1,000
g4	0,958	g10	0,583
g5	0,875	g11	0,708

Tabella 5.12: Accuracy per classe per il tipo di input depth map - IR - RGB.

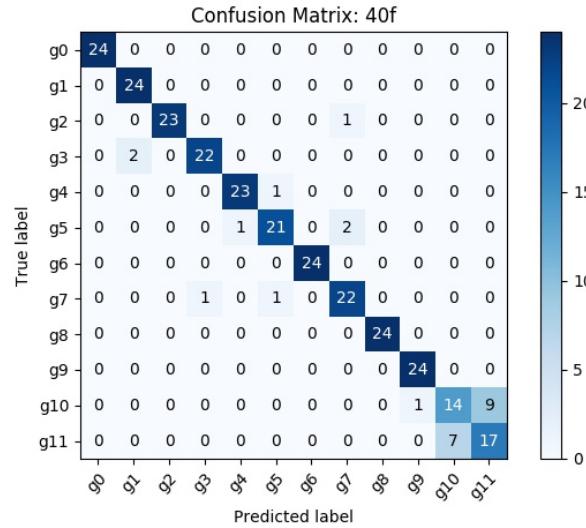


Figura 5.14: Matrice di confusione per il tipo di input: depth map - IR - RGB.

Accuracy per classe			
Arch. Multimod. - Depth Map, IR, RGB gs			
Class	Accuracy	Class	Accuracy
g0	1,000	g6	1,000
g1	1,000	g7	0,917
g2	0,917	g8	1,000
g3	1,000	g9	1,000
g4	0,958	g10	0,583
g5	0,875	g11	0,625

Tabella 5.13: Accuracy per classe per il tipo di input depth map - IR - RGB grayscale.

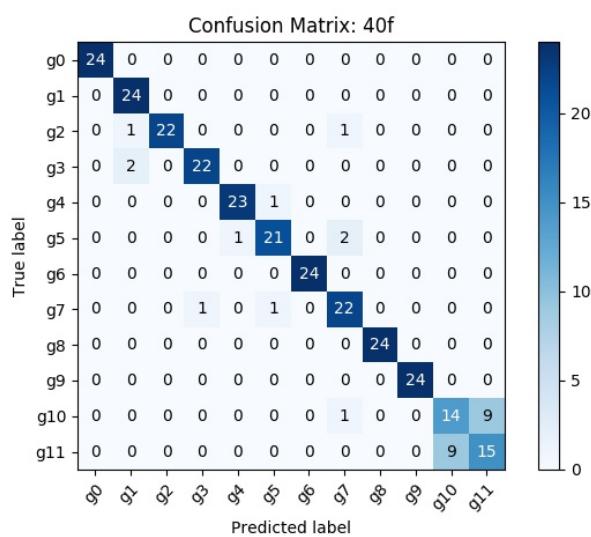


Figura 5.15: Matrice di confusione per il tipo di input: depth map - IR - RGB gs.

5.3.2 Analisi Risultati

Come si evince dalle metriche riportate il metodo migliore prevede la combinazione delle reti neurali convolutive (DenseNet) che processano input del tipo Depth Map e IR.

Inoltre è possibile notare come l'elemento che contribuisce ad una valutazione più migliore del metodo in fase di test sia l'utilizzo delle mappe di profondità, difatti, ove questo tipo di input è assente si ottengono i risultati peggiori. Difatti come già esposto per la valutazione del metodo precedente, tale risultato è dovuto al maggiore contenuto informativo insito nelle immagini di profondità. Difatti i gesti proposti essendo dinamici coinvolgono maggiormente le tre dimensioni spaziali aggiungendo dunque più informazione rispetto alle più statiche immagini 2D.

In contrapposizione, è evidente come l'utilizzo di immagini RGB non risulti particolarmente discriminante al fine del riconoscimento dei gesti. Anzi, è rilevante come l'uso di tale input contribuisca al peggioramento della precisione generale del metodo nel riconoscimento dei gesti. Infatti, come osservabile nella tabella 5.6, aggiungendo immagini RGB all'architettura multimodale composta dagli input depth map e immagini IR, si ottiene un decremento delle prestazioni. Si passa da un'accuracy media del 92% senza immagini RGB ad una del 91% con l'utilizzo di quest'ultime.

Inoltre, come per il metodo precedente (si veda 5.2), è possibile notare, analizzando le metriche di accuracy per classe delle varie modalità di input, come l'accuracy cali drasticamente per le classi g10 e g11 che vengono spesso confuse. Come affermato precedentemente, tale fenomeno negativo è dovuto alla notevole similarità del gesto compiuto che consiste in una rotazione dell'indice e del dito medio con pugno chiuso in senso orario per la classe g10 e in senso opposto per la classe g11.

5.4 Approccio con Reti Neurali Convolute 3D

5.4.1 Modalità di Training

In questa sezione sono riportati il procedimento e la configurazione di *training* utilizzati per il metodo proposto.

Per ogni tipo di dato, depth map, immagini infrarossi, immagini RGB e immagini RGB convertite in scali di grigi, è stato effettuato il training della rete per la singola configurazione di input (N, C_{in}, D, H, W) , con $C_{in} = 40$, poiché come evidenziato nei metodi proposti precedentemente garantisce un'accuratezza che non si discosta molto dalle altre configurazioni in termini di risultati e da un punto di vista computazionale richiedevano un training della rete esoso in termini di tempo di addestramento che comunque non avrebbero portato a miglioramenti rilevanti.

Per il training della rete sono stati utilizzati i pesi *pre-addestrati* sul dataset *Sports1M* [26], effettuando infine un *fine-tuning* per i layer modificati, esclusivamente per le modalità *depth maps* e IR. Per tutti gli altri tipi di immagine (RGB e RGB in scala di grigi) la rete è stata allenata *from scratch*, ovvero nella sua interezza inizializzandola con pesi casuali. Tale scelta è dovuta ai risultati sperimentali ottenuti in fase di training. Difatti è stato osservato come utilizzando pesi casuali in fase di inizializzazione ed effettuando un training completo della rete per i casi RGB e RGB in scala di grigi, la rete fornisse un'accuracy maggiore in *validation* rispetto alla rete con pesi *pre-addestrati*. Ciò invece non si è verificato in caso di utilizzo di immagini di profondità, di fatti tale configurazione è stata inizializzata con i pesi *pre-addestrati* sul dataset *Sports1M*.

I dati prima di essere forniti alla rete sono stati normalizzati sottraendo la media e dividendo per la varianza, calcolate per ogni modalità di immagine su tutto il *training set*, in modo da ottenere media nulla e varianza unitaria.

La rete è stata allenata per 200 epoche con *batch size* di 8 in caso di *depth map*, 4 per le altre modalità di immagine, scelta effettuata poiché in fase sperimentale si è ottenuta la migliore accuratezza del modello in fase di *validation* con tale configurazione, *learning rate* pari a 10^{-2} e *momentum* di 0.5. La funzione di *loss* utilizzata è la *binary categorical cross-entropy* e lo stochastic gradient descent (SGD) come ottimizzatore.

I risultati ottenuti in fase di *training* e *validation* per le diverse modalità sono visibili nei grafici a seguire.

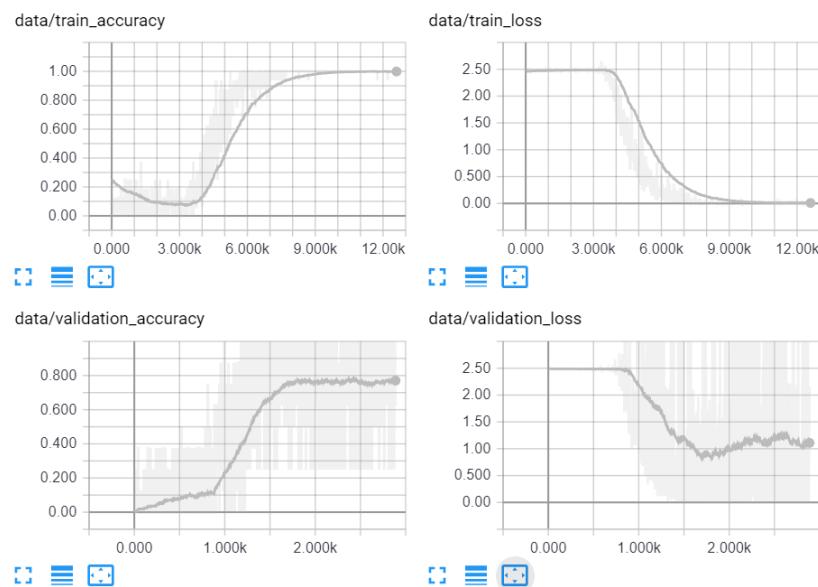


Figura 5.16: Input della rete: depth map. Grafico di accuracy e loss in *training* e *validation*

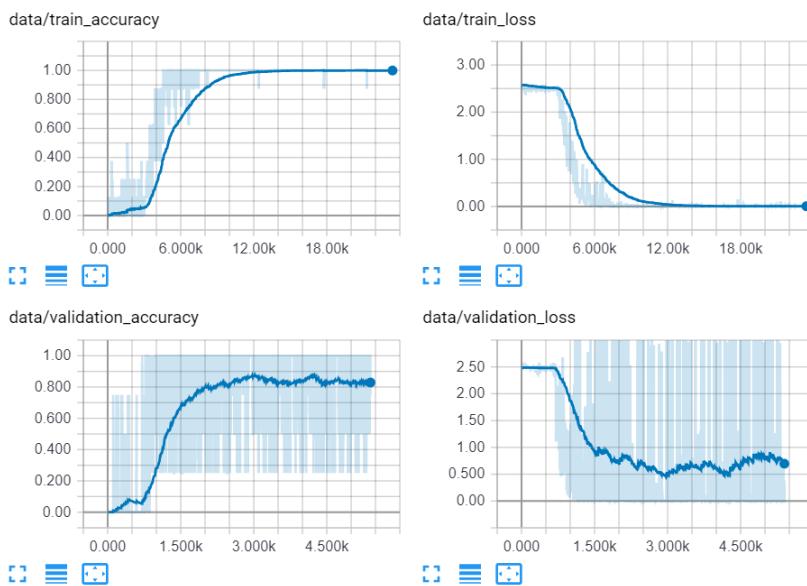


Figura 5.17: Input della rete: immagini IR. Grafico di accuracy e loss in *training* e *validation*

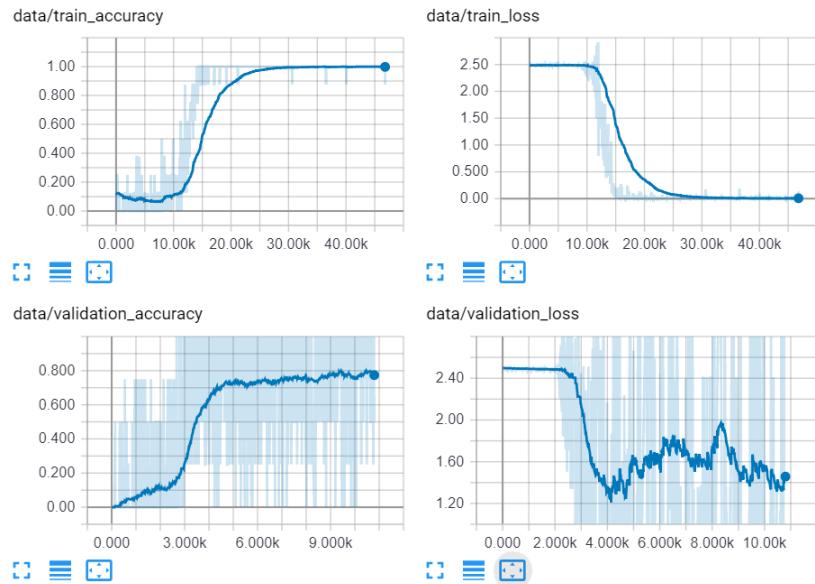


Figura 5.18: Input della rete: immagini RGB. Grafico di accuracy e loss in *training* e *validation*

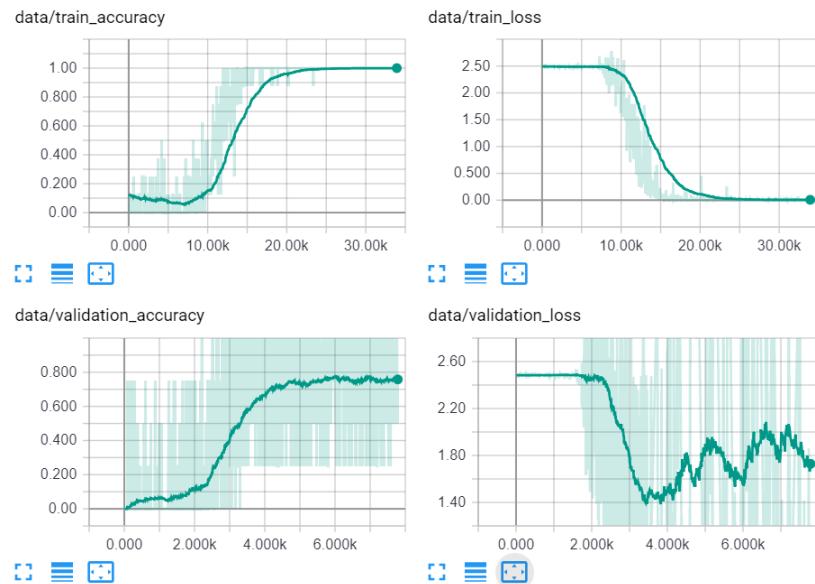


Figura 5.19: Input della rete: immagini RGB grayscale. Grafico di accuracy e loss in *training* e *validation*

5.4.2 Risultati

In questa sezione vengono presentati in maniera tabulare i risultati ottenuti nella valutazione del metodo proposto in 4.3 e le relative matrici di confusione. La valutazione è stato effettuato processando i gesti presenti nelle 8 sessioni del *dataset* dedicate al test, riproponendo la configurazione del metodo proposto in fase di training.

Le tipologie di dati utilizzate in fase di test per le quali sono riportate le relative metriche sono:

- **Depth map** (tab. 5.15 e fig. 5.20)
- **Immagini IR** (tab. 5.16 e fig. 5.21)
- **Immagini RGB** (tab. 5.17 e fig. 5.22)
- **Immagini RGB grayscale** (tab. 5.18 e fig. 5.23)

Il presente modello è stato valutato anch'esso esclusivamente con sequenze di lunghezza uguale a 40 *frame*. Infatti tale configurazione è stata adottata poiché come visto nelle sezioni precedenti ed in particolare nella tabella 5.1, i risultati al variare della lunghezza della sequenza non subiscono cambiamenti particolarmente rilevanti ai fini della precisione del metodo.

Accuracy C3D per tipo di input	
Input	Accuracy
Depth Map	0,760
Immagini IR	0,875
Immagini RGB	0,722
Immagini RGB Grayscale	0,633

Tabella 5.14: *Accuracy* media della rete C3D per ciascun tipo di input.

Successivamente sono riportate le tabelle di accuracy per classe [g0 ... g11] delle configurazioni di input riportate nella tabella 5.14 e le relative matrici di confusione. Il totale delle sequenze analizzate in fase di test è pari a 288, 24 sequenze per gesto.

Accuracy per classe C3D - Depth Map			
Class	Accuracy	Class	Accuracy
g0	0,708	g6	0,750
g1	0,875	g7	0,833
g2	0,750	g8	0,625
g3	0,792	g9	0,708
g4	0,833	g10	0,375
g5	0,917	g11	0,958

Tabella 5.15: Accuracy per classe per il tipo di input depth map.

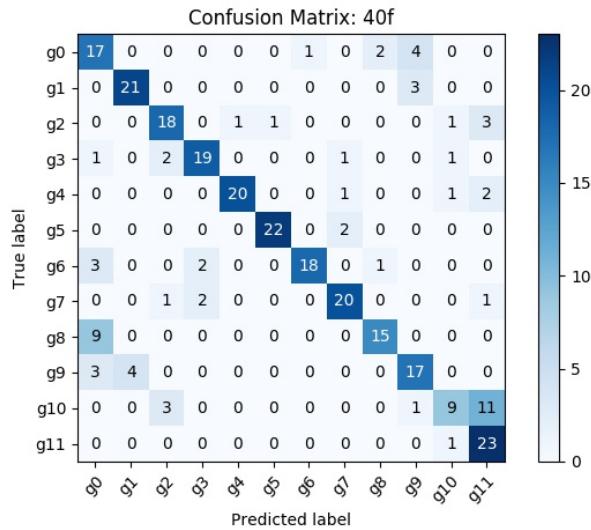


Figura 5.20: Matrice di confusione per il tipo di input: depth map.

Accuracy per classe C3D - IR			
Class	Accuracy	Class	Accuracy
g0	0,750	g6	0,958
g1	0,958	g7	0,875
g2	0,875	g8	1,000
g3	1,000	g9	1,000
g4	0,917	g10	0,750
g5	0,792	g11	0,635

Tabella 5.16: Accuracy per classe per il tipo di input immagini IR.

Accuracy per classe C3D - RGB			
Class	Accuracy	Class	Accuracy
g0	0,542	g6	0,917
g1	0,833	g7	0,750
g2	0,792	g8	0,917
g3	0,625	g9	0,667
g4	0,833	g10	0,542
g5	0,833	g11	0,417

Tabella 5.17: Accuracy per classe per il tipo di input immagini RGB.

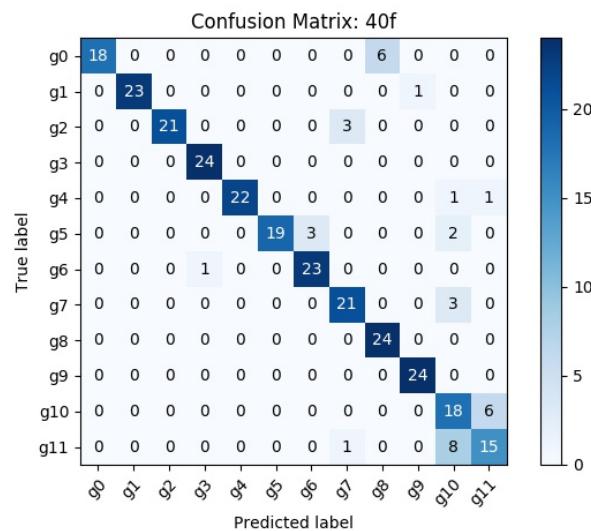


Figura 5.21: Matrice di confusione per il tipo di input: immagini IR.

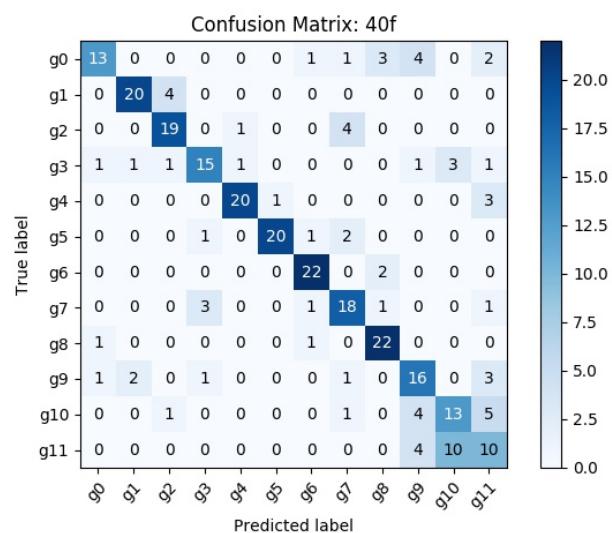


Figura 5.22: Matrice di confusione per il tipo di input: immagini RGB.

Accuracy per classe C3D - RGB gs			
Class	Accuracy	Class	Accuracy
g0	0,542	g6	0,750
g1	0,792	g7	0,458
g2	0,792	g8	0,708
g3	0,583	g9	0,583
g4	0,875	g10	0,583
g5	0,833	g11	0,458

Tabella 5.18: Accuracy per classe per il tipo di input immagini RGB grayscale.

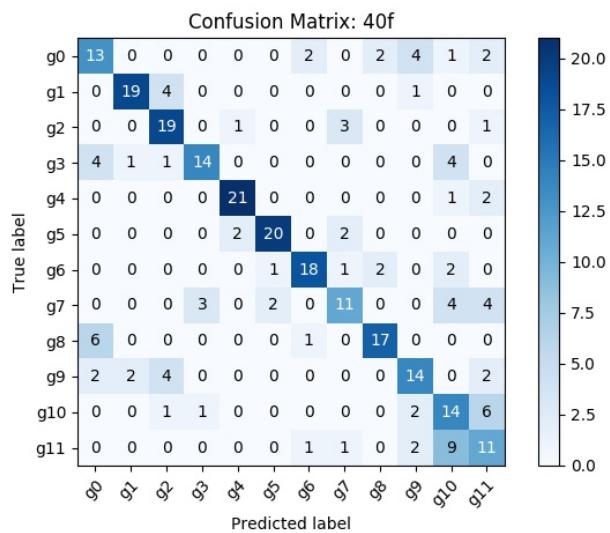


Figura 5.23: Matrice di confusione per il tipo di input: immagini RGB grayscale.

5.4.3 Analisi Risultati

I risultati riportati mostrano come la rete neurale convolutiva 3D, la C3D, riesca a fornire una prestazione decisamente più affidabile ed efficiente utilizzando immagini del tipo infrarossi (IR). Ciò era predicibile osservando l'andamento della funzione di accuracy in fase di training (si veda 5.17). Si nota come l'accuracy raggiunga valori più alti rispetto alle altre configurazioni. Le depth map, in contrapposizione ai risultati riportati dai metodi precedenti (si vedano le tabelle 5.1 e 5.6) non risultano la migliore modalità di input. In fase sperimentale infatti, la rete inizializzata con pesi casuali non ha dato segnali di apprendimento che dunque hanno costretto all'utilizzo dei pesi pre-addestrati sul dataset *Sports1M* [26]. Tali pesi però sono concepiti per essere utilizzati in caso di clip video RGB dunque non consoni per l'utilizzo di immagini di profondità.

Le immagini RGB e soprattutto quelle in scala di grigi permettono alla rete un riconoscimento dei gesti nettamente peggiore rispetto agli altri tipi di input proposti. Questo fenomeno è probabilmente dovuto al minore contenuto informativo apportato dalle immagini RGB per la loro scarsa qualità. Difatti, per emulare al meglio l'ambiente in cui il sistema dovrà svolgere la propria funzione, ovvero l'abitacolo di un veicolo, non sono state introdotte in fase di acquisizione del dataset fonti esterne di illuminazione. Questa scelta ha fatto sì che le immagini RGB presentino poco contrasto ed abbiano una scarsa luminosità.

Infine, in concordanza con i metodi già visti, si può notare come anche per la configurazione migliore le classi che ottengono valori di accuracy peggiori sono le classi g10 e g11, rappresentanti i due gesti dinamici più lunghi presenti nel dataset. Questo fenomeno è notabile attraverso la matrice di confusione riportata (fig. 5.21) e dalla tabella contenente l'accuracy per classe del metodo (tab. 5.16). È osservabile come l'accuracy cali per le classi g10 e g11 che, come riportato nella matrice di confusione, vengono spesso confuse. Ciò è dovuto come già esposto alla notevole similarità del gesto compiuto che consiste in una rotazione dell'indice e del dito medio con pugno chiuso in senso orario per la classe g10 e in senso opposto per la classe g11. Ne consegue che tali movimenti siano soggetti ad una male interpretazione da parte del sistema che comunque colpirebbe anche l'occhio umano.

5.5 Approccio con Reti Neurali Ricorrenti

5.5.1 Modalità di Training

In questa sezione sono riportati il procedimento e la configurazione di *training* utilizzati per il metodo proposto.

È stato effettuato il training della rete per le diverse configurazioni di input (*seq_len, batch, input_size*) con *seq_len* = 40, 35 e 30 poiché la tipologia di dato utilizzato non era stato analizzato nelle diverse configurazioni di lunghezza della sequenza.

La rete è stata allenata *from scratch*, ovvero nella sua interezza inizializzandola con pesi casuali.

I dati prima di essere forniti alla rete come input sono stati normalizzati sottraendo la media e dividendo per la varianza, calcolate il *training set*, in modo da ottenere media nulla e varianza unitaria.

La rete è stata allenata per 50 epoche con *batch size* di 2, *learning rate* pari a 10^{-3} e *weight decay* di 10^{-4} . La funzione di *loss* utilizzata è la *binary categorical cross-entropy* e adam [23] come ottimizzatore.

I risultati ottenuti in fase di *training* e *validation* per le diverse modalità sono visibili nei grafici a seguire.

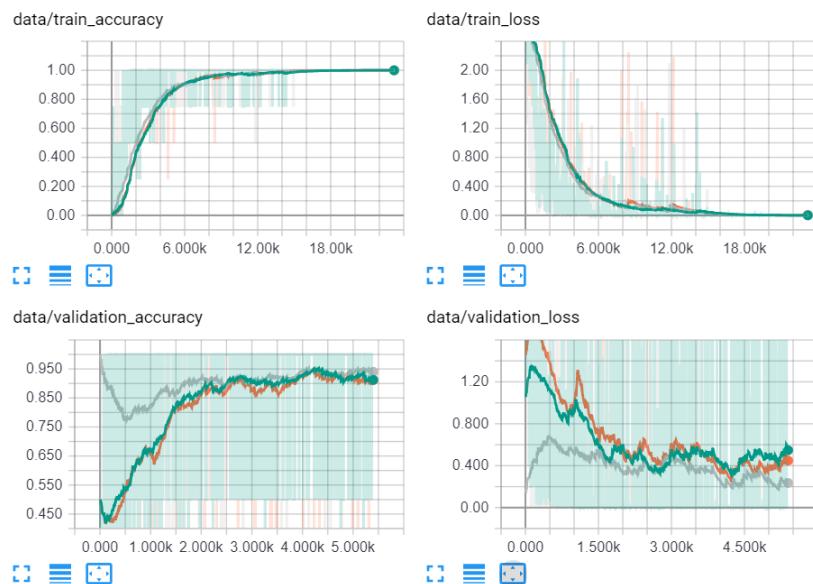


Figura 5.24: Coordinate 3D dei giunti: grafico di accuracy e loss in *training* e *validation* per N frame diversi: in arancio $\mathcal{N} = 40$; in verde $\mathcal{N} = 35$; in grigio $\mathcal{N} = 30$.

5.5.2 Risultati

In questa sezione vengono presentati in maniera tabulare i risultati ottenuti nella valutazione del metodo proposto in 4.4 (tab. 5.19).

La valutazione è stato effettuato processando i gesti presenti nelle 8 sessioni del *dataset* dedicate al test, riproponendo la configurazione del metodo proposto in fase di training.

LSTM - 3D Features	
Seq. len.	Accuracy
40	0.944
35	0.927
30	0.927

Tabella 5.19: Accuracy media del metodo LSTM al variare della lunghezza della sequenza.

Successivamente è riportata la tabella di accuracy per classe [g0 ... g11] per la configurazione di lunghezza della sequenza che ha ottenuto il miglior risultato (tab. 5.20) e la relativa matrice di confusione (fig. 5.25). Il totale delle sequenze analizzate in fase di test è pari a 288, 24 sequenze per gesto.

Accuracy per classe - LSTM - 3D features - 40			
Class	Accuracy	Class	Accuracy
g0	0,875	g6	1,000
g1	1,000	g7	0,917
g2	0,958	g8	0,875
g3	1,000	g9	1,000
g4	0,917	g10	0,917
g5	1,000	g11	0,875

Tabella 5.20: *Accuracy* per classe [g0 ... g11] della rete neurale ricorrente (LSTM) per il tipo di input: feature 3D. *Accuracy* calcolata per sequenze di lunghezza 40.

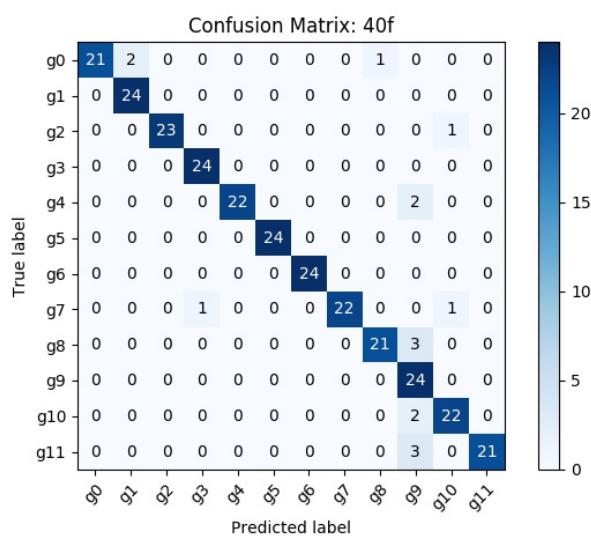


Figura 5.25: Matrice di confusione della rete LSTM Lunghezza sequenza = 40 frame.

5.5.3 Analisi Risultati

Le metriche riportate fanno emergere una certa omogeneità nelle prestazioni della rete al variare della lunghezza della sequenza. Tuttavia la sequenza più lunga raggiunge, anche se di poco, prestazioni leggermente più elevate.

In contrapposizione ai metodi proposti nelle sezioni precedenti, è possibile notare come l'*accuracy* si mantenga costante anche per le classi g10 e g11 che, come visibile nella matrice di confusione (tabella 5.25), vengono confuse nettamente meno rispetto ai metodi già visti. Ciò è probabilmente dovuto all'utilizzo del maggior contenuto informativo fornito dalle feature analizzate rispetto a quello presente nelle immagini. Mentre per le immagini il contenuto informativo è processato, appreso ed estratto dalle CNN che dunque può variare a seconda della qualità visiva del tipo di input fornito, nel caso del metodo corrente le informazioni vengono fornite nella loro versione definitiva e precisa, garantita dalla calibrazione ottimale del sensore su cui opera l'SDK.

Tuttavia, seppur in maniera non rilevante ai fini ultimi del funzionamento complessivo del sistema, si può osservare come le prestazioni peggiori si verifichino nel caso del riconoscimento delle classi g8 e g9, confrontandosi con i risultati ottenuti per le medesime classi dal metodo che utilizza le reti neurali convolutive con input *depth map*. Come osservabile nelle relative tabelle (tab. 5.2 e tab. 5.20), la rete neurale convolutiva per entrambe le classi ottiene il 100% dell'*accuracy* a differenza della rete neurale ricorrente che ottiene rispettivamente il 91,7% e l'87,5%. Tale fenomeno è causato dal movimento presente nelle classi g8 e g9. La chiusura della mano infatti risulta essere un movimento facile da riconoscere mediante l'utilizzo di immagini ma difficile da individuare per il sistema Leap Motion.

5.6 Conclusioni

In conclusione, nella seguente tabella (tab. 5.21) vengono riportate le metriche relative al miglior risultato ottenuto dai metodi, ciascuno relativamente alla configurazione di tipo di dato e lunghezza sequenza con cui il metodo ha ottenuto tale valutazione.

Accuracy			
CNN (Depth Map - 40)	Architettura Multimodale (Depth Map, IR - 40)	C3D (IR - 40)	LSTM (3D features - 40)
0,903	0,920	0,875	0,944

Tabella 5.21: Accuracy delle configurazioni migliori dei metodi proposti. Modello di rete; tipo di input; lunghezza sequenza in *frame*.

Come si evince dalle metriche riportate, la valutazione migliore in fase di test (94,4%) è ottenuta dall'uso della rete neurale ricorrente LSTM che opera sulle feature estrapolate dall' SDK del sensore Leap Motion (si veda il paragrafo 4.4.2). Inoltre, è possibile notare come la componente profondità giochi un ruolo fondamentale per la discriminazione delle varie *gesture*. Difatti, le reti neurali convolutive implementate, ad eccezione della C3D, ottengono i risultati migliori utilizzando proprio immagini di profondità, comunque presente anche nei giunti in input alla rete LSTM.

Infine Va sottolineato come il problema di accuracy minore e confusione per le classi g10 e g11 (rappresentanti i due gesti dinamici più lunghi presenti nel *dataset*) riscontrato nei metodi CNN, architettura multimodale e C3D, sia stato risolto utilizzando la rete ricorrente LSTM (si veda la matrice di confusione 5.25). Ciò è dovuto essenzialmente all'uso delle coordinate che contengono in se l'informazione di direzione del movimento a differenza della singola immagine sia essa di profondità, infrarossi o RGB.

Capitolo 6

Prototipo di sistema

Dopo aver effettuato i necessari esperimenti per determinare quale fosse la soluzione più accurata per la realizzazione di un sistema automatico per il riconoscimento di gesti, si è proceduto con l'implementazione del sistema in versione prototipale.

La scelta del metodo da utilizzare è ricaduta sull'approccio con CNN illustrato nella sezione 4.1, capitolo 4, con input depth map e sequenze di lunghezza 40.

Tale scelta è stata presa in considerazione del fatto che il suddetto metodo presenta un'affidabilità maggiore rispetto alla migliore soluzione fornita dalla rete LSTM con coordinate 3D dei giunti, nonostante le metriche rilevate siano a favore di quest'ultima (90,3% di *accuracy* contro 94,4%). Infatti il sensore utilizzato per l'acquisizione dei giunti ha diverse volte mostrato in fase di test problemi di affidabilità nel riconoscimento e tracciamento della mano in un contesto *real-time*. Inoltre è necessario sottolineare come l'utilizzo della rete CNN implementata sia indipendente dallo specifico sensore attraverso il quale vengono acquisite le immagini di profondità. Infatti, mentre è possibile riproporre lo stesso approccio architettonico delle CNN utilizzando un sensore diverso da quello utilizzato per l'acquisizione del *dataset*, risulta invece vincolante l'utilizzo del sistema Leap Motion per l'acquisizione dei giunti della mano per l'implementazione della rete LSTM proposta. Infine è importante evidenziare come con l'uso del Leap Motion siano necessarie delle informazioni note a priori, come la conoscenza della struttura della mano fornita dall' SDK proprietario. Tale necessità di informazioni complesse non risulta vincolante in caso di utilizzo di immagini che presentano un contenuto informativo diverso.

Inoltre, si è preferito utilizzare la singola rete neurale convolutiva con immagini di profondità invece dell'Architettura Multimodale con dati del tipo *depth map*

e immagini infrarossi pur avente *accuracy* migliore (92,0%), a discapito però di prestazioni computazionali (tempo di elaborazione) peggiori.

Il software prototipale è stato implementato utilizzando le librerie OpenCV e Numpy per il preprocessing delle immagini e la visualizzazione dell'interfaccia grafica, e PyTorch per l'utilizzo del metodo di riconoscimento di gesti scelto. Per la segmentazione temporale dei gesti è stato utilizzato un meccanismo a *sliding window* così funzionante: ogni immagine consecutiva acquisita dal sensore viene inserita in un *buffer*, al raggiungimento di una dimensione prefissata (40 *frames*), esso viene trasformato in tensore ed analizzato dalla rete neurale come input. Se la rete fornisce come risultato una classe [g0...g11] con affidabilità maggiore o uguale ad una soglia preimpostata (85% nel caso di studio), viene fornito in output l'identificativo del gesto corrispondente alla classe ed il *buffer* viene svuotato così che il procedimento possa iniziare nuovamente. Se la soglia non viene raggiunta il *frame* più vecchio viene sostituito con quello acquisito dal sensore correntemente e il *buffer* raggiunge nuovamente la dimensione necessaria per essere processato.

L'interfaccia grafica del software mostra i due flussi di immagine raccolti dal sensore Pico Flexx (immagini di profondità e infrarossi) e una *label* per informare se il sistema è in fase di acquisizione immagini (*Running*) o, in caso di avvenuta classificazione, per mostrare il gesto riconosciuto (*Detection*).

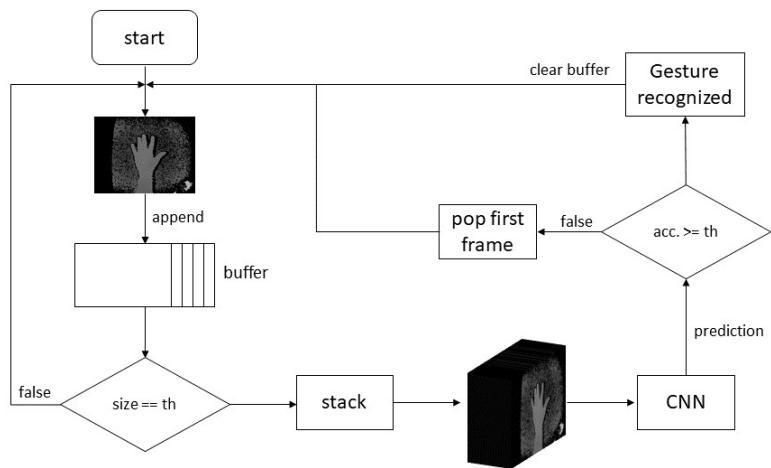


Figura 6.1: Diagramma di flusso del sistema prototipale.



Figura 6.2: Esempio di funzionamento del sistema prototipale: riconoscimento del gesto g00 - fist.

Capitolo 7

Conclusioni

Nell'ambito di questa relazione di tesi si è illustrata l'attività di ricerca volta allo sviluppo di un sistema per l'interazione uomo - veicolo tramite linguaggio naturale del corpo.

Il sistema si basa sul riconoscimento automatico di gesti mediante l'uso di sensori infrarossi e 3D ad opera di algoritmi di *deep learning* quali le reti neurali.

Si è illustrato come l'utilizzo di tecniche di *deep learning* necessiti di una grande mole di dati e di conseguenza, come si sia acquisito un *dataset* specifico per il caso di studio (contesto *automotive*) per far fronte a tale problema.

Sono state esposte tutte le procedure preliminari per la creazione di un *dataset ex novo*, l'ideazione, la configurazione dell'ambiente di acquisizione, la ricerca dei sensori idonei, la scrittura del software e infine la procedura di acquisizione vera e propria.

Sono stati studiati, analizzati e infine implementati diversi metodi di visione per il riconoscimento di gesti, ciascuno con la propria specificità e varietà nell'utilizzo dei dati presenti nel *dataset* acquisito.

Una volta implementati, i metodi sono stati sperimentati, valutati e comparati al fine di determinare la soluzione più efficiente ed idonea per realizzare il sistema di interazione uomo - veicolo.

Infine è stato realizzato un software demo che simuli il comportamento finale del sistema di riconoscimento per valutarne l'effettiva efficacia.

Molti possono essere gli sviluppi futuri del progetto di tesi esposto in questo elaborato. La reale applicazione del sistema HMI tramite riconoscimento di

gesti non è così lontana dalla realtà quotidiana che ci circonda. Sono già diffusi i primi sistemi che sfruttano questo tipo di interfaccia in veicolo seppur in maniera limitativa e poco efficiente.

Dunque, gli sviluppi futuri che si potranno avere, arricchiranno il panorama di interfacce naturali disponibili, col fine ultimo di facilitare l'interazione del guidatore con il proprio veicolo e non inficiare sulla sua sicurezza.

Capitolo 8

Ringraziamenti

Desidero innanzitutto ringraziare il Professore Vezzani per avermi dato la possibilità di conoscere in maniera più approfondita il settore della visione artificiale e la sua applicazione nel mondo reale, oltre la ricerca.

Ringrazio Guido Borghi e Stefano Pini per avermi guidato con esperienza e saggezza nel mio percorso di tesi, ed il gruppo del laboratorio per le serene giornate trascorse.

Devo molto alla mia famiglia, Papà, Mamma, Giuseppe, Chiara e Salvatore, che in più di cinque anni non hanno mai dubitato delle mie capacità e mi hanno sempre spronato a dare il meglio di me.

Ringrazio anche la mia famiglia modenese, lo zio Totò e la Zia Rosalba, secondi genitori ma non secondi in vicinanza e affetto. Casa loro è sempre stata casa mia.

Un grande pensiero va alle mie cugine Maria Luisa, Manuela e Paola, sorelle modenesi e sicuro appoggio.

Un ringraziamento speciale va a Giulio, fratello maggiore nella vita di tutti i giorni.

Infine ringrazio Stefano e Marco, coinquilini ma prima di tutto amici e compagni nelle mie avventure e disavventure quotidiane.

Bibliografia

- [1] Action recognition. <http://blog.qure.ai/notes/deep-learning-for-videos-action-recognition-review>. Last Accessed: 2019-03-11.
- [2] Automotive hmi, current status. http://www.aide-eu.org/pdf/final_workshop/day1/round_table/aide_day1_round_table_all.pdf. Last Accessed: 2019-03-11.
- [3] Automotive hmi redefined. https://www.eetimes.com/document.asp?doc_id=1272865&page_number=1. Last Accessed: 2019-03-11.
- [4] Autonui. https://www.researchgate.net/publication/233922479_AutoNUI_a_workshop_on_automotive_natural_user_interfaces. Last Accessed: 2019-03-11.
- [5] Deep learning. http://www.intelligenzaartificiale.it/deep-learning/#Cos8217e_il_deep_learning. Last Accessed: 2019-03-11.
- [6] Distracted driving | nhtsa. <https://www.nhtsa.gov/risky-driving/distracted-driving/>. Last Accessed: 2019-03-11.
- [7] Euroncap 2025 roadmap. www.euroncap.com/en/press-media/press-releases/euro-ncap-launches-road-map-2025-in-pursuit-of-vision-zero/. Accessed: 2019-03-11.
- [8] Final report: Study on the assessment and certification of automated vehicles. www.ec.europa.eu/growth/content/final-report-study-assessment-and-certification-automated-vehicles_en. Accessed: 2019-03-11.
- [9] Gpu nvidia. <https://www.nvidia.com/it-it/geforce/products/10series/geforce-gtx-1080/>. Last Accessed: 2019-03-11.
- [10] Incidenti stradali causa smartphone. <https://ilfattoquotidiano.it/incidenti-stradali-causa-distrazione-nel-mirino-gli-smartphone>. Last Accessed: 2019-03-11.
- [11] Leap motion. <https://www.leapmotion.com/>. Last Accessed: 2019-03-11.
- [12] Lstm, how it works. <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>. Last Accessed: 2019-03-11.

- [13] Lstm pytorch. <https://pytorch.org/docs/stable/nn.html>. Last Accessed: 2019-03-11.
- [14] Pico flexx. <https://pmdtec.com/picofamily/flexx/>. Last Accessed: 2019-03-11.
- [15] Pytorch. <https://pytorch.org/>. Last Accessed: 2019-03-11.
- [16] Rgb cam. <https://leopardimaging.com/product/li-ov5640-usb-72/>. Last Accessed: 2019-03-11.
- [17] Viva hand gesture challenge. <http://cvrr.ucsd.edu/vivachallenge/index.php/hands/hand-gestures/#cite1>. Last Accessed: 2019-03-11.
- [18] S. Y. Boulahia, E. Anquetil, F. Multon, and R. Kulpa. Dynamic hand gesture recognition based on 3d pattern assembled trajectories. In *2017 Seventh International Conference on Image Processing Theory, Tools and Applications (IPTA)*, pages 1–6. IEEE, 2017.
- [19] B. Boulay. *Human posture recognition for behaviour*. PhD thesis, Université Nice Sophia Antipolis, 2007.
- [20] L. Bretzner, I. Laptev, and T. Lindeberg. Hand gesture recognition using multi-scale colour features, hierarchical models and particle filtering. In *FGR*, 2002.
- [21] C. Craye and F. Karray. Driver distraction detection and recognition using rgb-d sensor. *arXiv preprint arXiv:1502.00250*, 2015.
- [22] P. M. X. Y. S. Gupta and K. K. S. T. J. Kautz. Online detection and classification of dynamic hand gestures with recurrent 3d convolutional neural networks. CVPR, 2016.
- [23] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.
- [24] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.
- [25] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.
- [26] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014.
- [27] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.
- [28] P. Molchanov, S. Gupta, K. Kim, and J. Kautz. Hand gesture recognition with 3d convolutional neural networks. pages 1–7, 06 2015.
- [29] S. S. Rautaray and A. Agrawal. Vision based hand gesture recognition for human computer interaction: a survey. *Artificial Intelligence Review*, 43(1):1–54, 2015.

- [30] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.
- [31] H. Sarbolandi, D. Lefloch, and A. Kolb. Kinect range sensing: Structured-light versus time-of-flight kinect. *Computer vision and image understanding*, 139:1–20, 2015.
- [32] M. Semprini. *Gesture Recognition: una panoramica*. PhD thesis.
- [33] A. A. Siddharth S. Rautaray. Vision based hand gesture recognition for human computer interaction: a survey. 2012.
- [34] M. Sundermeyer, R. Schlüter, and H. Ney. Lstm neural networks for language modeling. In *Thirteenth annual conference of the international speech communication association*, 2012.
- [35] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatio-temporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497, 2015.
- [36] Y. Wang, M. Huang, L. Zhao, et al. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 606–615, 2016.
- [37] Wikipedia. Radiazione infrarossa — wikipedia, l'enciclopedia libera, 2018. [Online; in data 20-marzo-2019].