



POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

TESI DI LAUREA MAGISTRALE IN
COMPUTER SCIENCE AND ENGINEERING -
INGEGNERIA INFORMATICA

AI for inclusivity: a deep learning approach for Italian Sign Language Recognition

Author:

Paolo Biolghini

Student ID: **246816**

Advisor: **Prof. Maristella Matera**

Co-advisor: **Ludovica Piro, Marco La Camera**

Academic Year: **2024-25**

Dedicated to my family.

Abstract

The inability of many people to understand sign language represents a significant communicative barrier for the Deaf community. Research on automatic sign language recognition aims to overcome this barrier, fostering more inclusive and effective communication. This study proposes a vision-based approach that relies on the extraction of skeletal keypoints, from which 42 joint angles are computed and used as input for the analyzed Machine Learning and Deep Learning models. In particular, the performance of 1D convolutional models, bidirectional LSTM architectures enhanced with attention mechanisms, and Transformer-based architectures has been evaluated. The work addresses three main tasks: isolated sign classification, out-of-distribution sign recognition, and continuous sign sequence classification. Experimental analyses were conducted on two private datasets containing 46 and 72 Italian Sign Language (LIS) signs, respectively.

Keywords: Italian Sign Language (LIS), 1D Convolutional Neural Network (1D-CNN), Bidirectional Long Short-Term Memory (Bi-LSTM), Transformer, isolated sign classification, out-of-distribution recognition, continuous sign sequence classification

Abstract in lingua italiana

L'incapacità di molte persone di comprendere la lingua dei segni rappresenta una significativa barriera comunicativa nei confronti della comunità dei sordi. Gli studi sul riconoscimento automatico della lingua dei segni mirano a superare questa barriera, promuovendo una comunicazione più inclusiva ed efficace. Questo studio propone un approccio *vision-based* basato sull'estrazione di punti chiave dello scheletro, dai quali vengono calcolati 42 angoli che costituiscono l'input per i modelli di *Machine Learning* e *Deep Learning* analizzati. In particolare, sono state valutate le prestazioni di modelli basati su convoluzioni 1D, architetture LSTM bidirezionali integrate con meccanismi di attenzione e architetture Transformer. Il lavoro si articola in tre attività principali: la classificazione di segni isolati, il riconoscimento di segni *Out-of-Distribution* e la classificazione di sequenze continue di segni. Le analisi sperimentali sono state condotte su due dataset privati contenenti rispettivamente 46 e 72 segni della Lingua dei Segni Italiana (LIS).

Parole chiave: Lingua Italiana dei Segni (LIS), 1D Convolutional Neural Network (1D-CNN), Bidirectional Long Short Term Memory (Bi-LSTM), Transformer, classificazione segni isolati, riconoscimento Out-of-Distribution, classificazione segni continui

Indice

Abstract	i
Abstract in lingua italiana	iii
Indice	v
1 Introduzione	1
1.1 Definizione del Problema	1
1.2 Contributo	2
1.3 Metodologia	2
1.4 Struttura della Tesi	3
2 Concetti Preliminari	5
2.1 Lingua dei Segni	5
2.1.1 Storia delle lingue dei segni e la nascita della LIS	6
2.1.2 Caratteristiche Linguistiche della Lingua dei Segni Italiana (LIS)	8
I parametri formazionali del segno	8
Le componenti non manuali	9
Peculiarità morfosintattiche e lessicali	9
2.2 Il Machine Learning nel Contesto della Traduzione Automatica	12
2.2.1 Definizione di Machine Learning	12
2.2.2 Categorie Principali di Apprendimento Automatico	13
2.2.3 Metodi e Modelli per problemi di Classificazione	13
2.2.4 Vantaggi e Sfide del Machine Learning	14
2.3 Deep Learning ed Architetture Neurali	14
2.3.1 La Struttura delle Reti Neurali Artificiali	15
2.3.2 Reti Neurali Convolutionali (CNN)	16
2.3.3 Reti Neurali Ricorrenti ed LSTM	17
2.3.4 Transformer ed Attention	17

2.3.5	Metriche per la Valutazione dei Modelli	19
	Metriche per il Riconoscimento di Segni Isolati	19
	Metriche per il Riconoscimento Continuo e la Traduzione	20
2.3.6	Rilevanza per la Traduzione della Lingua dei Segni Italiana	21
3	Stato dell'Arte	23
3.1	Tassonomia e Sfide Fondamentali	23
3.2	Pipeline di un Sistema di Traduzione	23
3.2.1	Inferenza in Tempo Reale	24
3.2.2	Addestramento del Modello	24
3.3	Metodologie di Acquisizione dei Dati	25
3.3.1	Approcci Sensor-based	25
3.3.2	Approcci Vision-based	27
	Approccio Frame Based	28
	Approccio Skeleton Based	28
	Approccio Ibrido	29
3.4	Modelli di Classificazione per l'ASLR	29
3.4.1	Classificazione di Dati da Sensori	30
3.4.2	Classificazione di Dati Basati sulla Visione	30
	Approcci Frame-based	31
	Approcci Skeleton-based	31
	Approcci Ibrido	32
3.4.3	Modellazione delle Dipendenze Temporali nel Riconoscimento Con- tinuo	33
3.5	Metriche di Valutazione	33
3.6	Risultati e Criticità	33
4	Metodi	37
4.1	Preparazione del Dataset	37
4.2	Classificazione di un singolo Segno	39
4.2.1	Individuazione delle Baseline	39
4.2.2	Sviluppo di Architetture di Deep Learning Avanzate	39
4.2.3	Utilizzo delle Funzioni di Loss	45
4.2.4	Strategie di Addestramento e Ottimizzazione degli Iperparametri	46
4.3	Out Of Distribution Detection	47
4.3.1	Approccio tramite Embedding con Anomaly Detection	48
4.3.2	Approccio basato sull'analisi dei logit	49
4.3.3	Approccio Teacher Student	50

4.4	Classificazione Multisegno	51
4.4.1	Baseline: Classificazione con Sliding Window	51
4.4.2	Secondo Approccio: Classificazione con Finestre di Dimensione Ri- dotta e <i>Major Voting</i>	52
4.4.3	Utilizzo di modelli LSTM end-to-end	53
5	Risultati	55
5.1	Risultati della Classificazione di Segni Singoli	55
5.1.1	Performance dei Modelli Baseline	56
5.1.2	Confronto tra Architetture di Deep Learning Avanzate	57
5.1.3	Impatto delle Funzioni di Loss	60
5.2	Risultati Out Of Distribution	62
5.2.1	CNN extraction + anomaly detection algorithms	62
5.2.2	Valutazione degli Approcci basati su Logit	64
5.2.3	Approcci Teacher-Student	65
5.3	Risultati della Classificazione Multisegno	67
5.3.1	Risultati del Metodo Baseline	67
5.3.2	Secondo Approccio: Classificazione con Finestre di Dimensione Ri- dotta e <i>Major Voting</i>	69
5.3.3	Performance dell'Approccio End-to-End	70
6	Discussione dei Risultati	71
6.1	Riconoscimento dei segni isolati	71
6.2	Rilevamento Out-Of-Distribution	72
6.3	Classificazione Multisegno	73
7	Conclusioni e Sviluppi Futuri	75
	Bibliografia	77
	Elenco delle figure	87
	Elenco delle tabelle	89
	Ringraziamenti	91

1 | Introduzione

Secondo un rapporto dell'Organizzazione Mondiale della Sanità (OMS), oltre 430 milioni di persone nel mondo soffrono di disabilità uditiva o del linguaggio e, di queste, l'80% è semi-analfabeta o analfabeta. Si stima inoltre che, entro il 2050, più di 700 milioni di persone, pari a una persona su dieci, presenteranno un deficit uditivo grave.[73] Queste cifre evidenziano l'esistenza di una vasta comunità globale per la quale la comunicazione verbale tradizionale può rappresentare una barriera sostanziale, rendendo necessario il ricorso a modalità comunicative alternative. In questo contesto, le lingue dei segni emergono come un mezzo di comunicazione fondamentale.

Sebbene persista una diffusa concezione errata, non esiste una lingua dei segni universale. Al contrario, una delle peculiarità fondamentali delle lingue dei segni risiede nel loro carattere fortemente regionale: esse presentano varianti e dialetti specifici a seconda dell'area geografica e della comunità di riferimento. È altresì importante sottolineare che la lingua dei segni non deve essere interpretata come un semplice strumento di comunicazione alternativa o subordinata rispetto alle lingue vocali, ma come un vero e proprio sistema linguistico autonomo, dotato di una propria grammatica, sintassi e lessico, e capace di esprimere in modo completo e articolato concetti complessi.

1.1. Definizione del Problema

Lo sviluppo di metodologie avanzate per il riconoscimento automatico delle lingue dei segni (ASLR) riveste un potenziale di rilevanza sociale straordinaria. Tali tecnologie, integrando tecniche di visione artificiale e modelli di apprendimento automatico, potrebbero abbattere le barriere comunicative che ancora oggi limitano la piena inclusione delle persone sorde in numerosi contesti, dall'istruzione ai servizi pubblici fino alla vita professionale. Inoltre, l'adozione di sistemi di traduzione automatica in tempo reale favorirebbe un'interazione più naturale e bidirezionale tra persone sorde e udenti, promuovendo una società maggiormente equa e inclusiva, in linea con gli obiettivi di accessibilità e integrazione sanciti a livello internazionale.

Dall'analisi della letteratura emerge come questo ambito di ricerca sia ancora in una fase di maturazione, con un crescente interesse negli ultimi anni ma con numerose sfide ancora aperte. In particolare, lo studio della Lingua dei Segni Italiana (LIS) risulta ostacolato dalla scarsità di risorse pubblicamente disponibili, che limita la possibilità di sviluppare modelli di apprendimento automatico su larga scala e di effettuare confronti sistematici tra approcci differenti.

1.2. Contributo

Nel presente lavoro è stata condotta un'analisi comparativa dei principali metodi per la classificazione della Lingua dei Segni Italiana (LIS), con particolare attenzione agli approcci basati su *Deep Learning*.

I contributi innovativi della tesi possono essere sintetizzati come segue:

- **Classificazione di segni isolati:** Confronto tra modelli di *Machine Learning* tradizionale e architetture di *Deep Learning* (CNN, LSTM, *Transformers*) per valutare la capacità di modellare le componenti spaziali e temporali del segnale.
- **Out-of-Distribution Detection:** Analisi della capacità dei modelli di distinguere tra segni appartenenti alla distribuzione di addestramento e segni inediti, utilizzando tecniche di *anomaly detection*, metodi logit-based e approcci *teacher-student*.
- **Classificazione di sequenze continue:** Sperimentazione di due strategie principali:
 - approccio a *sliding window* con *majority voting*;
 - soluzione *end-to-end* basata su reti LSTM per il riconoscimento di frasi composte da più segni consecutivi.
- **Ampiezza e generalità del dataset:** Tra i primi studi sul riconoscimento della LIS a utilizzare un *dataset* comprendente oltre 70 classi, ciascuna con più di 100 video per segno, caratterizzato da un'ampia copertura di termini di uso quotidiano senza un focus tematico specifico.

1.3. Metodologia

Il lavoro adotta un approccio *vision-based*, in particolare *skeleton-based*, utilizzando *MediaPipe* per estrarre informazioni relative a mani, volto e corpo dai video. Da queste informazioni sono stati calcolati 42 angoli caratteristici, impiegati come input per le diver-

se architetture. L'analisi sperimentale si basa su due *dataset* privati forniti dall'azienda Cloudia Research, contenenti lettere, numeri e parole di uso quotidiano: il primo include 46 segni, mentre il secondo ne comprende 72, per un totale complessivo di 9.700 video, con oltre 100 campioni per ciascun segno nel *dataset* di dimensioni maggiori.

1.4. Struttura della Tesi

Il presente lavoro è articolato secondo la seguente struttura:

- **Concetti Preliminari:** viene introdotta la Lingua dei Segni Italiana, la sua storia e la sua struttura, insieme a concetti fondamentali di *Machine Learning* (ML) e *Deep Learning*, elementi essenziali per lo studio del riconoscimento automatico della lingua dei segni.
- **Stato dell'arte:** viene analizzato come altri ricercatori hanno affrontato il problema, con approfondimenti sui metodi di acquisizione dei dati e sulle tecniche di classificazione nel contesto dell'ASLR.
- **Metodi:** viene descritta la preparazione dei dataset e le strategie adottate per affrontare le sfide della classificazione di segni isolati, della *Out-of-Distribution Detection* e della classificazione di sequenze multisegno.
- **Risultati:** per ciascun task e approccio vengono presentati i risultati sperimentali e le performance raggiunte.
- **Discussione:** i risultati ottenuti vengono confrontati criticamente e discussi, evidenziando punti di forza, limiti e implicazioni metodologiche.

2 | Concetti Preliminari

2.1. Lingua dei Segni

Le lingue dei segni non rappresentano semplicemente un'alternativa visiva alle lingue vocali, ma sono lingue visuo-gestuali autonome, capaci di veicolare contenuti complessi attraverso un sistema codificato di segni realizzati con le mani, espressioni facciali e movimenti del corpo.

Dal punto di vista linguistico, come affermato dal linguista Ferdinand de Saussure, la facoltà del linguaggio è indipendente dal canale fono-articolatorio [61]. Pertanto, l'uso del canale visivo-gestuale non comporta alcuna limitazione intrinseca alla complessità strutturale o alla capacità espressiva della lingua. Le lingue dei segni, infatti, possiedono una propria fonologia (cherologia), morfologia e sintassi, che le rendono complesse e complete tanto quanto le lingue orali.

Un elemento distintivo delle lingue dei segni è l'assenza, nella maggior parte dei casi, di un sistema di scrittura condiviso e ufficialmente codificato. Tale condizione, nota come agrafia, non è da intendersi come una carenza, bensì come una caratteristica comune a molte lingue naturali, soprattutto quelle trasmesse oralmente o visivamente all'interno di comunità ristrette. La trasmissione e l'evoluzione delle lingue dei segni avvengono dunque principalmente per via diretta e interpersonale, all'interno delle comunità segnanti.

L'utilizzo delle lingue dei segni non è esclusivo delle persone sorde. Non tutte le persone sorde, infatti, le impiegano nella comunicazione quotidiana, mentre molti individui udenti scelgono di apprenderle. Tra questi vi sono familiari di persone sorde, come genitori e figli, ma anche studenti, professionisti o semplici appassionati, attratti da motivazioni personali, culturali o professionali.

È inoltre importante ricordare che le lingue dei segni non sono nate unicamente per rispondere ai bisogni della comunità sorda. Esistono casi documentati di sistemi segnati sviluppati in contesti differenti: un esempio emblematico è rappresentato dalla lingua dei segni utilizzata dagli indigeni delle pianure nordamericane, impiegata come mezzo di

comunicazione interetnico tra gruppi che parlavano lingue vocali tra loro incomprensibili a causa delle marcate differenze dialettali.

Contrariamente a un'opinione diffusa, non esiste una lingua dei segni universale. Al contrario, vi sono numerose lingue dei segni nazionali e regionali, ciascuna con proprie regole grammaticali e lessicali. Tra le più note si annoverano la Lingua dei Segni Italiana (LIS), l'American Sign Language (ASL) e la British Sign Language (BSL). Anche all'interno di una stessa lingua nazionale si possono osservare significative variazioni dialettali. A differenza di quanto accade per le lingue vocali, i dialetti delle lingue dei segni non sono necessariamente distribuiti in base alla geografia, ma spesso rispecchiano la storia delle comunità segnanti, come quelle nate intorno a specifici istituti per sordi. Negli ultimi anni, tuttavia, si assiste a un processo di crescente standardizzazione, favorito dall'aumento della visibilità mediatica della LIS, attraverso televisione e social media, e dal ruolo attivo di centri culturali e formativi, che contribuiscono alla diffusione di una norma linguistica più unificata [70] [8].

2.1.1. Storia delle lingue dei segni e la nascita della LIS

L'interesse per le modalità comunicative delle persone sorde affonda le sue radici in epoche antiche. Nel Cratilo [54], Platone attribuisce alla gestualità una funzione espressiva intrinseca alla natura umana, mentre Aristotele [4, 5], pur distinguendo tra sordità e mutismo, alimenta una visione limitante, associando l'assenza di parola all'impossibilità di sviluppare pensiero astratto. La condizione sociale delle persone sorde risultava spesso marginalizzata, come evidenziato dal Codice di Giustiniano [22], che le classificava tra gli emarginati, privandole di diritti civili fondamentali, tra cui la possibilità di redigere un testamento o stipulare contratti.

Un primo, significativo cambiamento di prospettiva si ebbe durante il Rinascimento, con la nascita di un approccio strutturato per l'educazione dei sordi. Figure come il medico Girolamo Cardano teorizzarono che la vista potesse sopperire alla mancanza dell'udito, dimostrando che l'apprendimento non era indissolubilmente legato al canale acustico-vocale. Questi principi trovarono applicazione pratica grazie a pionieri come il monaco spagnolo Pedro Ponce de León (1520-1584), che sviluppò metodi educativi per i figli sordi di famiglie aristocratiche, insegnando loro a leggere, scrivere e contare attraverso tecniche che integravano la dattilologia (alfabeto manuale). Tuttavia, questi primi metodi erano spesso custoditi gelosamente come segreti professionali, limitandone la diffusione.

La svolta decisiva avvenne nel XVIII secolo in Francia, con il contributo determinante dell'abate Charles-Michel de l'Épée (1712-1789). A differenza dei suoi predecessori, de

L'Épée adottò un approccio pubblico e inclusivo, fondando a Parigi il primo istituto per sordi accessibile anche alle classi meno abbienti. Il suo "metodo dei segni metodici" consisteva in una combinazione dei segni naturali usati dai suoi studenti e di segni artificiali creati per rappresentare elementi della grammatica francese, come articoli e preposizioni. L'apertura del suo istituto non solo formò studenti sordi, ma contribuì anche alla preparazione di nuovi educatori, favorendo così la rapida diffusione del "metodo francese" in tutta Europa e negli Stati Uniti.

In Italia, questo modello pedagogico fu importato da figure chiave come l'abate Tommaso Silvestri(1744-1789), che nel 1784, dopo un periodo di formazione a Parigi, fondò la prima scuola per sordi a Roma. Poco dopo, l'abate Ottavio Assarotti(1753-1829) aprì un istituto a Genova, anch'esso basato sull'uso dei segni e della dattilologia per l'insegnamento. Nel corso del XIX secolo, sorsero circa cinquanta istituti in tutta Italia, creando contesti in cui le comunità sorde potevano ricevere un'istruzione di base e imparare mestieri utili all'inserimento nel mondo del lavoro.

Tuttavia, questo periodo di progresso subì una drastica interruzione con il Congresso di Milano del 1880. Durante questo evento, una mozione approvata a larga maggioranza sancì la superiorità del metodo oralista puro, bandendo di fatto le lingue dei segni dall'educazione formale con lo slogan "il gesto uccide la parola". Le ragioni di questa scelta furono diverse e riconducibili a motivazioni politiche e pedagogiche: da un lato, la necessità di promuovere l'unificazione linguistica dell'Italia post-unitaria, dall'altro l'influenza della scuola tedesca, secondo cui la lingua dei segni non sarebbe stata in grado di esprimere concetti astratti. Una conseguenza diretta fu l'emarginazione degli insegnanti sordi, che furono relegati a ruoli secondari nei laboratori artigianali degli istituti. Nonostante la messa al bando ufficiale, la lingua dei segni sopravvisse all'interno dei convitti e nelle nascenti associazioni per sordi.

Il XX secolo fu caratterizzato da una lenta ma progressiva riconquista dei diritti. Nacquero le prime associazioni di mutuo soccorso, che confluirono nel 1932 nell'Ente Nazionale Sordomuti (ENS), oggi Ente Nazionale Sordi. Fu però solo nella seconda metà del secolo, sulla scia delle ricerche pionieristiche condotte negli Stati Uniti da William Stokoe, che si iniziò a studiare la lingua dei segni da una prospettiva linguistica. In Italia, i primi gruppi di ricerca sorsero tra la fine degli anni '70 e l'inizio degli anni '80, portando alla definizione del termine Lingua dei Segni Italiana (LIS). Questa denominazione fu scelta per sottolineare come essa sia una lingua a tutti gli effetti, con una propria struttura grammaticale e sintattica, distinta sia dalla gestualità generica sia dalla lingua italiana parlata. [70] [8]

Nonostante questi progressi scientifici, l'Italia è stata uno degli ultimi Paesi europei a riconoscere giuridicamente, a livello nazionale, la propria lingua dei segni. Questo traguardo è stato raggiunto solo il 19 maggio 2021, con l'approvazione del Decreto Legge n. 41 [1]. L'articolo 34-ter del decreto afferma: "La Repubblica riconosce, promuove e tutela la lingua dei segni italiana (LIS) e la lingua dei segni italiana tattile (LIST)". La stessa norma riconosce formalmente anche le figure professionali dell'interprete LIS e LIST, figure chiave per l'inclusione delle persone con disabilità uditiva. [14]

2.1.2. Caratteristiche Linguistiche della Lingua dei Segni Italiana (LIS)

Le lingue dei segni, tra cui la Lingua dei Segni Italiana (LIS), sono sistemi linguistici completi, dotati di una grammatica autonoma e di una struttura articolata che si distingue nettamente da quella delle lingue vocali. La loro modalità espressiva si fonda su un canale visivo-gestuale, organizzato su due livelli principali e interdipendenti: le componenti manuali e le componenti non manuali.

I parametri formazionali del segno

Così come le parole nelle lingue vocali possono essere scomposte in fonemi, anche i segni sono analizzabili in unità più piccole, denominate parametri formazionali o cheremi. L'identificazione di questi parametri si basa sul principio della coppia minima: una variazione in uno solo di essi comporta un cambiamento di significato.

I principali parametri manuali sono quattro:

- **Luogo di articolazione:** indica il punto del corpo o dello spazio in cui viene eseguito il segno. Ad esempio, il segno *MAMMA* si articola sulla guancia, mentre altri segni si realizzano nello spazio neutro davanti al busto. Vi sono segni che si distinguono esclusivamente per questo parametro, come *CONOSCERE* (alla tempia) e *PARLARE* (alla bocca).
- **Configurazione della mano:** si riferisce alla forma assunta dalla mano durante il segno e varia in base all'inventario specifico di ogni lingua dei segni. Una differenza nella configurazione può modificare radicalmente il significato, come nei segni *BICICLETTA* e *CAMBIARE*.
- **Movimento:** descrive l'azione compiuta dalla o dalle mani e può variare per direzione, intensità e tipo (lineare, circolare, ondulatorio). Anche questo parametro consente di distinguere segni simili in ogni altro parametro, come *PREPARARE* e

FRESCO.

- **Orientamento del palmo:** indica la direzione verso cui è rivolto il palmo durante l'esecuzione del segno. Ad esempio, *STUDIARE* e *PORTA* differiscono proprio in questo aspetto.

Le componenti non manuali

Oltre ai parametri manuali, le lingue dei segni si avvalgono delle componenti non manuali, che svolgono un ruolo essenziale nel conferire senso, coerenza e funzione all'enunciato. Questi elementi, attivi in simultanea con quelli manuali, veicolano informazioni grammaticali, sintattiche e pragmatiche imprescindibili.

L'espressione facciale è fondamentale per segnalare la modalità della frase. Le domande chiuse si accompagnano a sopracciglia sollevate, quelle aperte richiedono sopracciglia aggrottate mentre il tono imperativo si esprime attraverso una configurazione facciale specifica. Lo sguardo ha una funzione deittica e anaforica, poiché stabilisce relazioni tra i segni e i referenti nello spazio, indica la direzione dell'azione nei verbi direzionali e regola i turni conversazionali. I movimenti della bocca si distinguono in due categorie: la labializzazione, che consiste nell'articolazione silenziosa di parole vocali e i morfemi orali, ovvero movimenti della bocca non associati a parole, dotati di valore grammaticale e in grado di modificare il significato di un segno. Ad esempio, un rigonfiamento della guancia può distinguere il segno *LAVORO* 2.1 da *PRESTITO* 2.2. La postura del corpo, infine, contribuisce a organizzare la struttura del discorso e a delimitare lo spazio sintattico. Nei contesti narrativi, essa permette l'impersonamento dei personaggi, attraverso cui il segnante assume fisicamente il ruolo dei protagonisti della narrazione.

Peculiarità morfosintattiche e lessicali

La LIS presenta caratteristiche morfosintattiche e lessicali che la rendono profondamente diversa dalle lingue vocali.

Un primo elemento distintivo è l'equilibrio tra iconicità e arbitrarietà. Molti segni sono iconici, cioè rappresentano visivamente il loro referente, come nel caso di *CASA* 2.4 o *TELEFONO* 2.5. Tuttavia, l'iconicità non implica una trasparenza universale: ciascuna lingua dei segni seleziona e convenzionalizza aspetti specifici del referente, rendendo il segno comprensibile solo dopo che ne è noto il significato. Per questo motivo, il segno per *CANE* in LIS si distingue da quello in ASL e BSL, come illustrato in figura 2.7. In ogni caso, l'iconicità coesiste sempre con l'arbitrarietà, ovvero l'assenza di un legame naturale tra segno e significato. Un altro aspetto centrale è l'uso dello spazio sintattico. Lo spazio

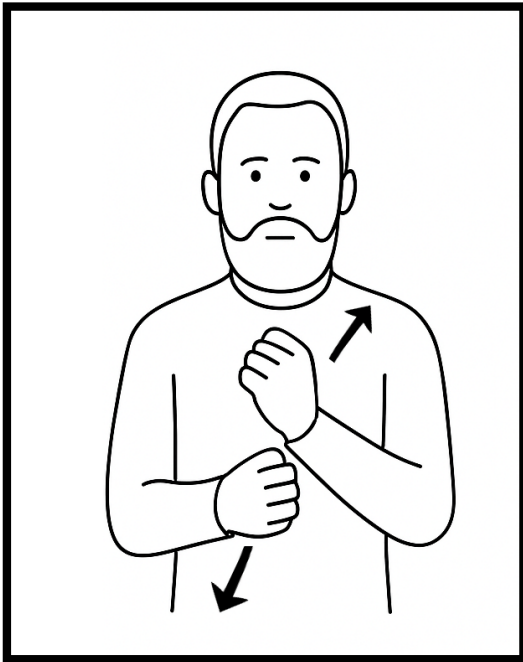


Figura 2.1: Segno "Lavoro"

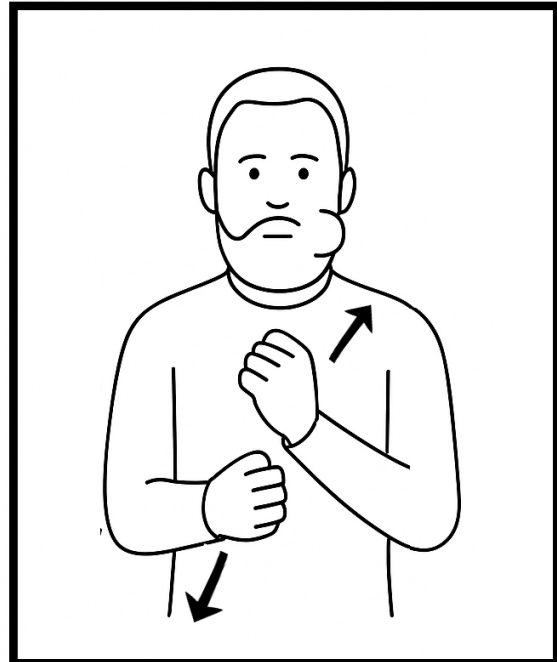


Figura 2.2: Segno "Prestito"

Figura 2.3: Segni "Lavoro" e "Prestito" della LIS. Illustrazioni realizzate dall'autore a partire da materiali presenti in [64].

di fronte al segnante assume un valore grammaticale tridimensionale: i referenti vengono collocati in posizioni specifiche e possono essere richiamati successivamente tramite un semplice gesto direzionale. Questo meccanismo, noto come accordo spaziale, consente ai verbi di flettersi per soggetto e oggetto. Ad esempio, nel verbo DARE, il movimento indica la direzione dell'azione da chi compie l'atto a chi lo riceve, come in IO-DARE-A-TE rispetto a TU-DARE-A-ME.

La simultaneità è una caratteristica fondamentale delle lingue dei segni: a differenza delle lingue vocali, in cui i fonemi vengono prodotti in sequenza, i parametri di un segno si realizzano contemporaneamente, consentendo di veicolare una grande quantità di informazione in un singolo gesto. Questa proprietà permette di combinare espressione e descrizione in maniera sinergica, soprattutto nelle strutture narrative, attraverso l'impiego di tecniche di trasferimento:

- **Trasferimento di forma:** consente di rappresentare oggetti attraverso la configurazione e il movimento delle mani.
- **Trasferimento di situazione:** descrive azioni o processi, con lo sguardo del segnante rivolto verso la scena spaziale creata.

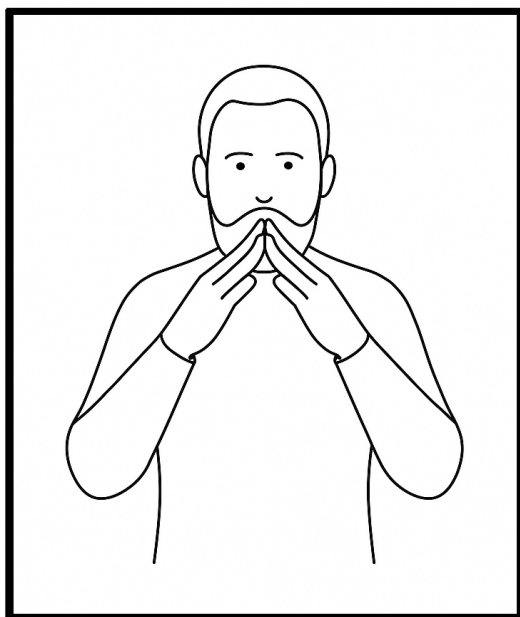


Figura 2.4: Segno "Casa"

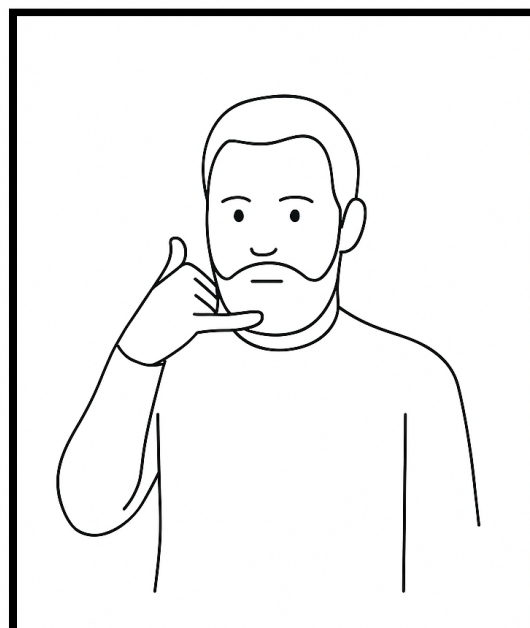


Figura 2.5: Segno "Telefono"

Figura 2.6: Segni "Casa" e "Telefono" della LIS. Illustrazioni realizzate dall'autore a partire da materiali presenti in [64].

- **Trasferimento di persona** (o impersonamento): permette al segnante di assumere completamente il ruolo del personaggio, replicandone espressioni, postura e azioni.

Accanto ai segni convenzionali presenti nei dizionari, il lessico della LIS include una componente altamente produttiva: i classificatori. Si tratta di configurazioni della mano che rappresentano categorie di referenti (ad esempio, persone, veicoli, oggetti piatti) e che possono essere manipolate nello spazio per descrivere posizione, movimento e interazione in modo flessibile e iconico.

Infine, la LIS, come tutte le lingue dei segni, è soggetta a una significativa variabilità e dinamicità: nuovi segni nascono per riflettere cambiamenti culturali e tecnologici, come avviene nel caso del segno per TELEFONO. Inoltre, sono presenti segni derivanti da altre lingue dei segni, come il segno COMUNICARE, di origine statunitense, oggi ampiamente diffuso tra i giovani segnanti italiani. [70] [8]

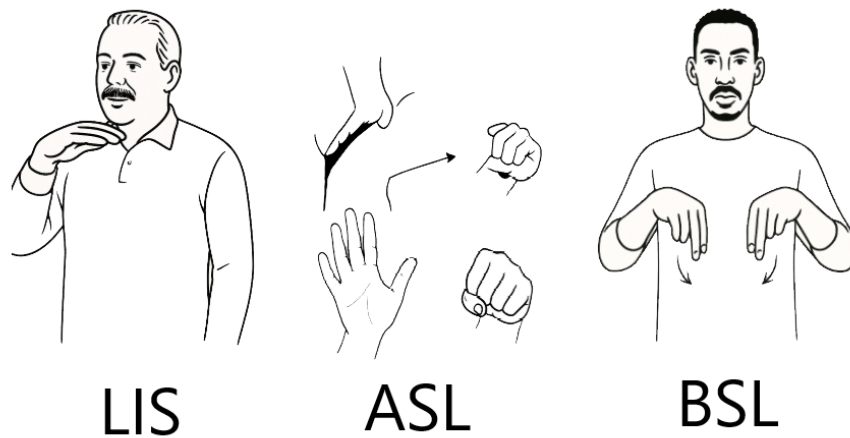


Figura 2.7: Il segno per "cane" in tre lingue dei segni: LIS (Lingua dei Segni Italiana), ASL (American Sign Language) e BSL (British Sign Language). L'immagine evidenzia come ciascuna lingua performi il segno in maniera differente. Illustrazioni realizzate dall'autore a partire da materiali presenti in [64].

2.2. Il Machine Learning nel Contesto della Traduzione Automatica

Nell'odierno panorama tecnologico e scientifico, il *Machine Learning* (ML) rappresenta una delle discipline a più rapida crescita e diffusione. La sua importanza scaturisce dalla crescente capacità di generare, raccogliere e archiviare enormi volumi di dati, i quali, se analizzati con metodi tradizionali, rivelerebbero solo una frazione del loro potenziale informativo. Il *Machine Learning* offre gli strumenti per esplorare questi vasti *dataset*, scoprendo *pattern* complessi e correlazioni nascoste che superano le capacità dell'analisi statistica convenzionale. Questa caratteristica rende il machine learning una componente essenziale in un'ampia gamma di applicazioni che spaziano dall'elaborazione del linguaggio naturale alla visione artificiale, fino alla previsione di fenomeni complessi.

2.2.1. Definizione di Machine Learning

Il *Machine Learning* è un sottoinsieme dell'Intelligenza Artificiale (AI) focalizzato sulla creazione di sistemi capaci di apprendere e migliorare le proprie prestazioni attraverso l'esperienza, senza essere stati programmati esplicitamente per ogni singolo compito. Il concetto di ML venne introdotto per la prima volta da Arthur Samuel nel 1959, che lo descrisse come la possibilità per un calcolatore di acquisire capacità di apprendimento senza istruzioni dettagliate per ogni attività specifica [60]. Successivamente, Tom Mitchell

ha proposto una definizione più rigorosa e ampiamente utilizzata in ambito ingegneristico:

Un programma informatico si dice che apprenda dall'esperienza E rispetto a un certo compito T e a una misura di prestazione P se le sue prestazioni su T , misurate da P migliorano con l'esperienza E [48].

Generalmente il processo di apprendimento si articola tipicamente in tre fasi: un processo decisionale genera una previsione basata sui dati di *input*, una funzione di errore valuta l'accuratezza di tale previsione; e un processo di ottimizzazione modifica il modello per ridurre la discrepanza tra la stima e il risultato atteso, iterando questo ciclo fino al raggiungimento di un livello di accuratezza desiderato.

2.2.2. Categorie Principali di Apprendimento Automatico

Gli approcci del Machine Learning possono essere classificati in tre categorie principali, che pur non essendo le uniche esistenti, rappresentano le più rilevanti. Esse si distinguono in base alla natura dei dati e al tipo di feedback fornito al sistema durante l'addestramento. L'apprendimento supervisionato (*supervised learning*) è il più comune e si basa sull'utilizzo di *dataset* etichettati, in cui a ogni dato di *input* è associato il corrispondente *output* corretto. L'algoritmo apprende a stabilire una corrispondenza funzionale tra *input* e *output*, con l'obiettivo di generalizzare tale conoscenza a dati non visti; un esempio classico è il filtraggio delle email di *spam*. Al contrario, l'apprendimento non supervisionato (*unsupervised learning*) opera su dati non etichettati, cercando di identificare autonomamente strutture intrinseche, come nel caso del *clustering* di clienti con comportamenti d'acquisto simili per analisi di mercato. Infine, l'apprendimento per rinforzo (*reinforcement learning*) si distingue per un approccio basato su tentativi ed errori, in cui un agente impara a compiere azioni in un ambiente per massimizzare una ricompensa cumulativa; questo paradigma è impiegato, tra le altre applicazioni, nell'addestramento di sistemi capaci di giocare a scacchi. [3]

2.2.3. Metodi e Modelli per problemi di Classificazione

Nell'ambito dell'apprendimento automatico, i problemi di classificazione sono affrontati attraverso una vasta gamma di modelli computazionali. Tra gli approcci fondazionali si annovera la *Logistic Regression*, un modello lineare ampiamente utilizzato per la sua efficienza e interpretabilità in problemi di classificazione binaria e multiclasse. Accanto ad essa, altre metodologie classiche come i classificatori *Naive Bayes*, basati sull'applicazione del teorema di Bayes, o il *K-Nearest Neighbors* (K-NN), un approccio non parametrico basato sull'istanza, hanno storicamente fornito solide baseline per numerose applicazioni.

Procedendo verso modelli di maggiore complessità, un ruolo di primo piano è ricoperto dai *decision trees*, che partizionano lo spazio delle caratteristiche mediante una struttura gerarchica di regole. Una loro diretta evoluzione, le *random forests*, mitiga il rischio di overfitting e migliora la robustezza del classificatore aggregando le previsioni di un insieme di alberi. Un'altra metodologia di riferimento è costituita dalle *Support Vector Machines* (SVM), algoritmi particolarmente potenti che operano costruendo un iperpiano ottimale in uno spazio ad alta dimensionalità, con l'obiettivo di massimizzare il margine tra le diverse classi di dati. Di particolare rilevanza per il dominio applicativo di questa tesi sono le Reti Neurali Artificiali (Artificial Neural Networks, ANN). Le architetture di Deep Learning, che si fondano su reti neurali con molteplici strati, rappresentano lo stato dell'arte in questo campo. La loro capacità di apprendere gerarchie di *features* direttamente dai dati grezzi le rende eccezionalmente performanti in compiti caratterizzati da elevata complessità, come il riconoscimento di immagini e l'elaborazione del linguaggio naturale, discipline fondamentali per la traduzione automatica della Lingua dei Segni. [3, 43]

2.2.4. Vantaggi e Sfide del Machine Learning

I vantaggi offerti dall'adozione di tecniche di *Machine Learning* sono molteplici e includono l'automazione di processi complessi e ripetitivi, l'elevata capacità di adattamento a nuovi dati e il miglioramento continuo delle *performance* nel tempo. I modelli di ML possono analizzare volumi di dati altrimenti intrattabili, fornendo analisi predittive che supportano decisioni strategiche. Tuttavia, l'applicazione di questi strumenti comporta sfide significative. La qualità dei dati è fondamentale, poiché dati di scarsa qualità, incompleti o distorti portano inevitabilmente a risultati inaffidabili, secondo il principio "*garbage in, garbage out*". Un'altra problematica rilevante è il *bias*, ovvero la tendenza di un modello a replicare e amplificare pregiudizi sistemici presenti nei dati di addestramento. Altre sfide includono il rischio di *overfitting*, ovvero un'eccessiva aderenza ai dati di *training* che compromette la capacità di generalizzazione, e la scarsa interpretabilità di alcuni modelli complessi che rendono difficile comprendere il processo decisionale dell'algoritmo. [3]

2.3. Deep Learning ed Architetture Neurali

Il *Deep Learning* (DL) rappresenta una branca del *Machine Learning* che si basa sull'impiego di reti neurali artificiali con molteplici strati di elaborazione (*layers*). Questa profondità architeturale consente ai modelli di apprendere gerarchie di caratteristiche

(*features*) a livelli di astrazione crescenti, partendo da dati grezzi. Tale capacità si è rivelata fondamentale per risolvere problemi di elevata complessità, come quelli legati alla percezione visiva e alla comprensione del linguaggio naturale. [3]

2.3.1. La Struttura delle Reti Neurali Artificiali

Le reti neurali artificiali (ANN, Artificial Neural Networks) sono modelli computazionali ispirati al funzionamento sinaptico del cervello umano [44, 57]. Sono composte da unità fondamentali chiamate *neuroni*, organizzate in strati (*layers*) distinti: uno strato di input, uno o più strati nascosti (*hidden layers*) e uno strato di output.

Ogni neurone riceve in input un insieme di valori (provenienti dai neuroni dello strato precedente), che vengono combinati linearmente tramite pesi (w) e bias (b), e successivamente trasformati da una funzione di attivazione non lineare ϕ [23]. L'output del neurone è quindi calcolato come:

$$y = \phi \left(\sum_i w_i x_i + b \right)$$

Le funzioni di attivazione più comuni includono la *ReLU* (Rectified Linear Unit), la *sigmoide* e la *tangente iperbolica*, ciascuna con vantaggi specifici in termini di capacità espressiva e comportamento del gradiente [49]:

$$\text{ReLU}(x) = \max(0, x) \quad \sigma(x) = \frac{1}{1 + e^{-x}} \quad \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

L'apprendimento si realizza attraverso un processo iterativo di *backpropagation* [58], in combinazione con un algoritmo di ottimizzazione, generalmente rappresentato dalla discesa del gradiente (*gradient descent*). Durante l'addestramento, l'errore prodotto in uscita dalla rete (rispetto al valore atteso) viene propagato a ritroso attraverso gli strati, aggiornando i pesi in modo da minimizzare una funzione obiettivo (*loss function*), come ad esempio la *cross entropy* per problemi di classificazione [23].

L'impiego di molteplici strati nascosti conferisce alle reti neurali la capacità di apprendere rappresentazioni gerarchiche dei dati [37]: gli strati iniziali catturano caratteristiche di basso livello (ad esempio contorni in immagini), mentre quelli finali apprendono concetti progressivamente più astratti e discriminativi.

2.3.2. Reti Neurali Convoluzionali (CNN)

Le *Convolutional Neural Networks* (CNN) rappresentano una classe di reti neurali particolarmente efficaci nell'elaborazione di dati con struttura spaziale, come le immagini [35]. A differenza delle reti neurali completamente connesse (fully connected), le CNN sfruttano la correlazione locale dei dati, riducendo drasticamente il numero di parametri e migliorando l'efficienza computazionale.

[23] Una CNN è composta da una serie di strati specializzati, tra cui:

- **Strati convoluzionali (convolutional layers):** applicano filtri (*kernel*) che operano localmente su porzioni limitate dell'immagine, estraendo caratteristiche come bordi, angoli o texture. Ogni filtro genera una *mappa di attivazione* (feature map) che evidenzia la presenza di specifici pattern [63].
- **Funzioni di attivazione:** applicate elemento per elemento alle mappe di attivazione prodotte dalle operazioni di convoluzione, introducono non linearità nel modello. La funzione più comunemente utilizzata anche in questo caso è la *ReLU* [49].
- **Strati di pooling:** riducono la dimensionalità spaziale delle feature map, mantenendo le informazioni più rilevanti e riducendo il rischio di overfitting [35]. Le operazioni più comuni sono:

$$\text{MaxPooling : } y_{i,j} = \max_{(m,n) \in \mathcal{R}_{i,j}} x_{m,n}$$

$$\text{AveragePooling : } y_{i,j} = \frac{1}{|\mathcal{R}_{i,j}|} \sum_{(m,n) \in \mathcal{R}_{i,j}} x_{m,n}$$

dove $\mathcal{R}_{i,j}$ rappresenta la regione di pooling centrata nella posizione (i, j) dell'input x .

- **Strati completamente connessi:** presenti generalmente nella parte finale della rete, trasformano le *features* estratte in predizioni finali.

Il principale vantaggio delle CNN risiede nella loro capacità di apprendere automaticamente rappresentazioni gerarchiche e invarianti alle trasformazioni locali, come traslazioni, rotazioni o variazioni di scala [67]. Questo le rende ideali per compiti di riconoscimento visivo, come la classificazione di immagini, il riconoscimento di oggetti e, nel caso specifico della LIS, il riconoscimento dei segni espressi visivamente.

L'addestramento delle CNN segue lo stesso schema delle reti neurali standard, mediante l'ottimizzazione iterativa dei pesi attraverso la *backpropagation*[58], ma con l'aggiunta

della condivisione dei pesi tra neuroni convoluzionali, che consente di catturare pattern ricorrenti nell'intera immagine con un numero contenuto di parametri.

2.3.3. Reti Neurali Ricorrenti ed LSTM

Le *Reti Neurali Ricorrenti* (RNN, *Recurrent Neural Networks*) [13, 59] sono architetture progettate per l'elaborazione di dati sequenziali, in cui l'ordine temporale degli input è rilevante. A differenza delle reti feedforward, le RNN mantengono una memoria interna che consente di modellare dipendenze temporali, aggiornando il proprio stato nascosto (*hidden state*) ad ogni passo temporale in funzione dell'input corrente e dello stato precedente.

Tuttavia, le RNN tradizionali presentano alcune limitazioni significative, tra cui i problemi di *vanishing gradient* e *exploding gradient*, che ostacolano l'apprendimento di dipendenze a lungo termine. Tali problemi derivano dal ripetuto utilizzo della stessa funzione ricorrente nel tempo, che può causare l'amplificazione o l'annullamento dei gradienti durante la retropropagazione [7, 26]. Per superare queste difficoltà, sono state introdotte le *Long Short-Term Memory* (LSTM) [21, 26], un tipo di RNN progettato per conservare e aggiornare informazioni rilevanti nel tempo tramite un meccanismo di memoria esplicito. Come illustrato in fig. 2.8, ogni unità LSTM è composta da uno stato della cella (*cell state*) che funge da "memoria a lungo termine", e da tre *gate* principali:

- **Input gate** (i_t): controlla quali informazioni dell'input corrente devono essere aggiunte allo stato della cella.
- **Forget gate** (f_t): decide quali informazioni dello stato precedente devono essere rimosse.
- **Output gate** (o_t): regola quali informazioni della cella vengono trasferite all'output e allo stato nascosto.

Questi meccanismi permettono alla rete di apprendere selettivamente le informazioni da mantenere, dimenticare o trasmettere, favorendo l'apprendimento di dipendenze a lungo raggio anche in sequenze molto lunghe.

2.3.4. Transformer ed Attention

Il meccanismo di *attention* [6, 41] ha rivoluzionato il campo dell'apprendimento automatico per sequenze, offrendo un'alternativa più efficiente e flessibile rispetto alle reti neurali ricorrenti. Alla base di questo paradigma si trova l'idea che, durante l'elaborazione di un elemento di output, il modello debba potersi "focalizzare" selettivamente su parti

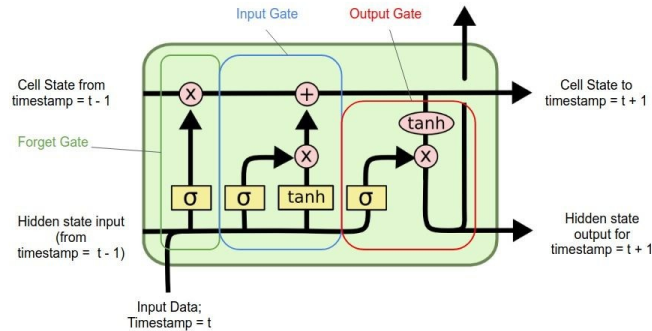


Figura 2.8: Schema di una cella LSTM, con i tre gate principali: *input*, *forget* e *output*, e il flusso delle informazioni attraverso lo stato di cella e lo stato nascosto.

specifiche dell'input, attribuendo loro un peso relativo in base alla loro rilevanza. Questo approccio è particolarmente utile nei compiti *sequence-to-sequence*, dove ogni elemento dell'output può dipendere da diverse parti della sequenza di input.

Le architetture *Transformer*, introdotte da Vaswani et al. (2017) [69], si basano interamente sul meccanismo di attenzione, eliminando la dipendenza da strutture ricorrenti. Il Transformer utilizza la *self-attention* per modellare le relazioni tra tutti gli elementi della sequenza in parallelo. A ogni passo, il modello calcola una rappresentazione pesata dell'intera sequenza, che permette di catturare contesti globali in modo più efficiente rispetto alle RNN o LSTM.

Il cuore del meccanismo di attenzione è la funzione:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

dove Q , K e V sono rispettivamente le matrici delle *query*, *key* e *value*, e d_k è la dimensione delle chiavi. Questo approccio consente al modello di confrontare ogni elemento della sequenza con tutti gli altri per determinare su quali concentrarsi.

L'assenza di ricorrenza nei Transformer permette inoltre una maggiore parallelizzazione durante l'addestramento e una migliore gestione delle dipendenze a lungo termine. Queste caratteristiche li rendono oggi una delle architetture più avanzate per il trattamento di sequenze complesse.

Nel contesto del riconoscimento della Lingua dei Segni Italiana (LIS), il meccanismo di attention si rivela particolarmente efficace per individuare i frame più salienti all'interno di una sequenza video. Ad esempio, quando un modello deve interpretare un segno, non tutti i frame hanno la stessa importanza: il meccanismo di attention consente di assegnare

maggiore rilevanza ai momenti chiave del gesto, migliorando la capacità del modello di comprendere la struttura e il significato del segno.

2.3.5. Metriche per la Valutazione dei Modelli

La valutazione delle prestazioni di un modello di ML o di DL rappresenta un passaggio cruciale nel processo di sviluppo e validazione. Durante la fase di addestramento e, soprattutto, in fase di test, è fondamentale disporre di metriche quantitative che permettano di misurare in modo oggettivo l'efficacia del modello nel risolvere il compito assegnato. Tali metriche non solo consentono il confronto tra diversi modelli o configurazioni, ma forniscono anche indicazioni preziose per l'ottimizzazione dell'architettura e per l'individuazione di fenomeni indesiderati, come l'overfitting o il bias.

Metriche per il Riconoscimento di Segni Isolati

Nel contesto del riconoscimento di segni isolati, dove ogni input corrisponde a una singola classe predefinita, il problema è valutato come un classico compito di classificazione multiclasse. Le metriche più comuni sono:

Accuracy È la metrica più diffusa e intuitiva. Misura la percentuale di segni classificati correttamente sul totale delle predizioni. Sebbene di facile interpretazione, può risultare fuorviante in presenza di dataset sbilanciati, dove alcune classi sono molto più rappresentate di altre. Il termine *Recognition Rate*, frequentemente incontrato in letteratura, è nella maggior parte dei casi un sinonimo di accuratezza.

Per un'analisi più dettagliata, spesso ottenuta a partire dalla matrice di confusione, si ricorre alle seguenti metriche:

- **Precision:** misura, per una data classe, la proporzione di predizioni corrette rispetto a tutte le volte che il modello ha predetto quella classe. Risponde alla domanda: "Quando il modello prevede il segno *casa*, quante volte ha ragione?".
- **Recall :** misura, per una data classe, la proporzione di istanze correttamente identificate rispetto a tutte le reali istanze di quella classe. Risponde alla domanda: "Di tutti i segni *casa* presenti nel dataset, quanti ne ha individuati il modello?".
- **F1-Score:** è la media armonica di Precision e Recall e fornisce un singolo valore di sintesi che bilancia entrambi gli indicatori. È particolarmente utile in caso di sbilanciamento tra le classi.

Metriche per il Riconoscimento Continuo e la Traduzione

Quando il sistema deve trascrivere o tradurre un'intera sequenza di segni, che forma una frase, le metriche di classificazione singola risultano inadeguate. In questo scenario, è necessario valutare la sequenza prodotta nel suo complesso, tenendo conto dell'ordine delle parole e di eventuali errori.

Word Error Rate (WER) È la metrica standard *de facto* per questo compito, mutuata dal campo del riconoscimento automatico del parlato (ASR). Il WER calcola la distanza di editing tra la sequenza di parole predetta dal modello e la sequenza di parole di riferimento (la trascrizione corretta). Si calcola come:

$$\text{WER} = \frac{S + D + I}{N}$$

dove S è il numero di *sostituzioni*, D il numero di *delezioni*, I il numero di *inserzioni* e N il numero totale di parole nella sequenza di riferimento

Un valore di WER più basso indica una performance migliore.

BLEU (Bilingual Evaluation Understudy) In alcuni studi, soprattutto quelli che si configurano come veri e propri sistemi di traduzione automatica da lingua dei segni a lingua parlata, viene impiegata anche la metrica BLEU. Originaria del campo della traduzione automatica, BLEU misura la somiglianza tra la frase prodotta dal modello e una o più traduzioni di riferimento, calcolando la precisione degli *n-grammi* (sottosequenze di parole) e introducendo una penalità per le frasi troppo corte.

$$\text{BLEU} = \text{BP} \cdot \exp \left(\sum_{n=1}^N w_n \log p_n \right)$$

dove p_n è la precisione degli n -grammi, w_n è il peso associato a ciascun ordine di n -gramma (tipicamente $w_n = \frac{1}{N}$), e BP è il *brevity penalty*, definito come:

$$\text{BP} = \begin{cases} 1, & \text{se } c > r, \\ e^{(1-r/c)}, & \text{se } c \leq r, \end{cases}$$

con c lunghezza della frase candidata e r lunghezza della frase di riferimento più vicina.

Un punteggio BLEU più alto corrisponde a una traduzione migliore.

2.3.6. Rilevanza per la Traduzione della Lingua dei Segni Italiana

I concetti esposti sono direttamente pertinenti al presente lavoro di tesi, che si propone di applicare il *Machine Learning* alla traduzione automatica della Lingua dei Segni Italiana (LIS). Nello specifico, il progetto sfrutta modelli di *Deep Learning*, basati su reti neurali profonde, per analizzare sequenze video di segni e tradurle in testo scritto. Questo compito richiede la capacità di apprendere rappresentazioni complesse e dinamiche dai dati visivi, un'area in cui le reti neurali eccellono. La traduzione della LIS rappresenta una sfida complessa che risulta particolarmente adatta all'approccio supervisionato, nel quale i modelli vengono addestrati su un *corpus* di video etichettati con la corrispondente traduzione testuale. Le tecniche di ML, quindi, non sono solo uno strumento, ma il fondamento metodologico che permette di affrontare un problema di frontiera con l'obiettivo di sviluppare una tecnologia ad alto impatto sociale e inclusivo.

3 | Stato dell'Arte

Il campo del riconoscimento automatico della lingua dei segni (ASLR) ha conosciuto un'evoluzione significativa a partire dagli anni '90. Tale crescita ha subito una notevole accelerazione negli ultimi anni, grazie ai significativi progressi nei settori del *machine learning* e della *computer vision*.

3.1. Tassonomia e Sfide Fondamentali

In letteratura, i sistemi di ASLR vengono generalmente classificati in base a diverse dimensioni: la natura del segno (manuale o non manuale), la tipologia (statica o dinamica), e la modalità di analisi (isolata o continua). La distinzione tra riconoscimento isolato e continuo è particolarmente rilevante; nel riconoscimento isolato ogni sequenza di input rappresenta un singolo segno, facilitando il processo di classificazione. Al contrario, il riconoscimento continuo implica l'elaborazione di sequenze contenenti più segni consecutivi, spesso privi di chiari confini temporali tra l'uno e l'altro. Questo rende il problema sensibilmente più complesso, anche per la sua natura intrinsecamente *weakly supervised*: risulta infatti difficile, se non impossibile, fornire etichette di addestramento che siano perfettamente allineate nel tempo con l'esecuzione dei gesti. Di conseguenza, i modelli affrontano una duplice sfida: non solo riconoscere i singoli segni, ma anche segmentare correttamente le sequenze temporali e gestire le ambiguità legate alla variabilità individuale nell'esecuzione dei segni.

3.2. Pipeline di un Sistema di Traduzione

Per affrontare le sfide di classificazione e segmentazione temporale finora descritte, lo sviluppo e l'impiego di un sistema di ASLR seguono un flusso operativo strutturato, comunemente definito "pipeline". Tale architettura processuale può essere analizzata scomponendola in due fasi distinte ma complementari: la pipeline di inferenza, che descrive il funzionamento del sistema in tempo reale, e la pipeline di addestramento, che ne governa la creazione e l'ottimizzazione.



Figura 3.1: Struttura tipica per la traduzione automatica in tempo reale della lingua dei segni, articolato in fasi di acquisizione, elaborazione e classificazione del segnale gestuale.

3.2.1. Inferenza in Tempo Reale

La pipeline tipica per la traduzione automatica della lingua dei segni in tempo reale si articola in diverse fasi sequenziali. Quando un segno viene eseguito dalla persona segnan- te, il gesto viene acquisito mediante un'interfaccia uomo-macchina, la quale converte il movimento in un segnale digitale. Quest'ultimo viene pre-processato per poi essere fornito in input a un algoritmo di classificazione che ha il compito di identificare il segno corri- spondente. In alcuni casi, al termine della fase di classificazione, è previsto un ulteriore modulo di conversione che consente di trasformare il segno riconosciuto in testo scritto o parlato, come avviene nel sistema proposto in [75].

3.2.2. Addestramento del Modello

Complementare alla fase di inferenza è il processo di addestramento del modello, che costituisce un passaggio cruciale per le prestazioni del sistema. Questo ha inizio con la creazione di un nuovo *dataset* o, in alternativa, con l'adozione di un *dataset* già esistente. I dati acquisiti devono essere opportunamente etichettati, in modo da permettere un apprendimento supervisionato efficace.

Segue quindi una fase di preprocessing, che può includere operazioni di pulizia del segnale, normalizzazione e riduzione della dimensionalità, con l'obiettivo di migliorare la qualità dei dati e facilitare l'apprendimento. A questa possono essere affiancate proce- dure di segmentazione e tracciamento della regione d'interesse, volte a isolare le compo- nenti significative del gesto. Successivamente, si procede all'estrazione delle *features*, che rappresentano l'informazione rilevante da fornire al modello.

Infine, si passa alla fase di addestramento vera e propria, durante la quale il modello di

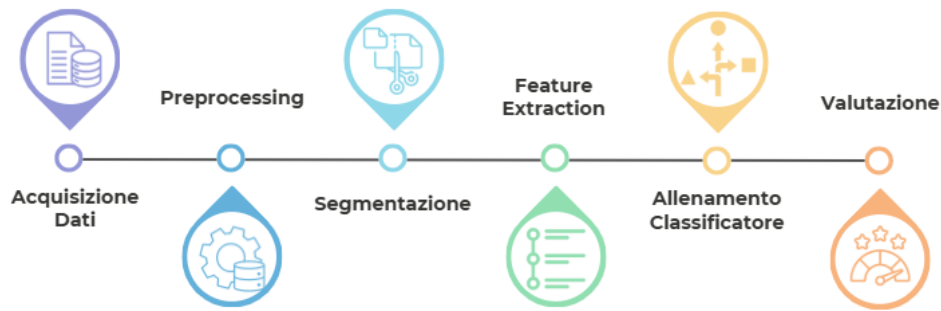


Figura 3.2: Pipeline del processo di addestramento in un sistema di riconoscimento dei segni: dalla raccolta dei dati, passando per le fasi di preprocessing e di estrazione delle caratteristiche, fino all’addestramento e alla valutazione del modello.

classificazione apprende la mappatura tra le caratteristiche estratte e i segni corrispondenti. Il ciclo si conclude con una rigorosa fase di valutazione tramite apposite metriche, volta a misurare le performance e a guidare eventuali cicli di ottimizzazione, culminando in un modello validato e pronto per l’impiego [42].

3.3. Metodologie di Acquisizione dei Dati

I metodi di acquisizione dei dati per l’ASLR possono essere ricondotti a due macro-categorie: approcci basati su sensori (sensor-based) e approcci basati sulla visione (vision-based).

3.3.1. Approcci Sensor-based

Gli approcci sensor-based si fondano sull’impiego di dispositivi dotati di sensori, spesso integrati in supporti indossabili come guanti intelligenti, in grado di rilevare in modo diretto il movimento e la postura delle mani. L’idea di utilizzare guanti sensorizzati per interpretare i gesti risale agli anni ’80, quando si iniziò a esplorare il potenziale dell’interazione gestuale tra esseri umani e computer. Nel 1983, Gary Grimes, ingegnere presso i Bell Labs, progettò un guanto in grado di riconoscere i 26 gesti manuali dell’alfabeto manuale americano, impiegato nella American Sign Language, con l’obiettivo di facilitare l’inserimento di dati attraverso i gesti [15]. Qualche anno dopo, nel 1988, i ricercatori della Stanford University James Kramer e Larry Leifer presentarono il primo dispositivo concepito esplicitamente per agevolare la comunicazione tra persone sorde e udenti: il cosiddetto talking glove.[34]

Tra gli approcci più promettenti, si annoverano quelli che sfruttano materiali con particolari proprietà meccaniche ed elettriche, capaci di tradurre le deformazioni fisiche in

segnali elettrici grazie all'effetto piezoresistivo: la variazione di resistenza elettrica generata dalla sollecitazione meccanica consente infatti di ottenere un segnale proporzionale allo sforzo applicato [75].

La letteratura recente evidenzia tuttavia un panorama molto più ampio, con diverse linee di ricerca che esplorano l'integrazione di materiali avanzati, sensori multipli e algoritmi di apprendimento automatico. Ad esempio, alcuni lavori hanno introdotto guanti basati su nanogeneratori triboelettrici (TENG), in grado di alimentarsi autonomamente e di trasformare i movimenti biomeccanici delle dita in segnali elettrici per il riconoscimento della lingua dei segni, riducendo la dipendenza da fonti di alimentazione esterne e migliorando la robustezza del dispositivo [33].

Un'altra direzione di ricerca è rappresentata da guanti realizzati con materiali compositi e sensori distribuiti, come nel caso del "Deep learning-powered composite foam smart glove" [24], dove l'integrazione di modelli di *deep learning* consente il riconoscimento in tempo reale di gesti complessi. Analogamente, un sistema sviluppato per la lingua dei segni tanzaniana utilizza un guanto dotato di sensori inerziali (IMU) e pieghevole, combinando algoritmi di fusione dei dati per ottenere un'accuratezza del 96,5% su 26 segni distinti [10].

Sono stati inoltre proposti prototipi a basso costo che cercano un compromesso tra accuratezza, comfort e accessibilità economica, come il Dataglove for Sign Language Recognition (2023), che utilizza molteplici sensori inerziali per riconoscere gesti su persone con diversa manualità [29]. Infine, il lavoro "Machine Learning-Based Gesture Recognition Glove" (2024) ha impiegato cinque sensori di flessione, cinque sensori di forza e un IMU, insieme a una rete neurale convoluzionale, per riconoscere gesti dinamici non appartenenti a nessuna lingua dei segni oltre alle posture statiche, raggiungendo un'accuratezza prossima al 90% [18].

Questo paradigma presenta vantaggi significativi, quali un'elevata accuratezza nell'acquisizione dei dati, una scarsa sensibilità ai disturbi ambientali e, soprattutto, una ridotta necessità di effettuare complesse operazioni di *feature extraction*. A partire dall'inizio del XXI secolo, tali sistemi hanno conosciuto una crescente diffusione, come dimostrano i numerosi premi ricevuti da diversi progetti di ricerca. Tra questi, il caso che ha ottenuto maggiore rilevanza mediatica è stato SignAloud, un prototipo sviluppato dagli studenti della University of Washington, Thomas Pryor e Navid Azodi [55]. Il successo di iniziative simili ha inoltre favorito la nascita di alcuni tentativi di commercializzazione, grazie alla relativa semplicità dei modelli computazionali impiegati.

Tuttavia, gli approcci *sensor-based* presentano notevoli limitazioni. In primo luogo, l'e-

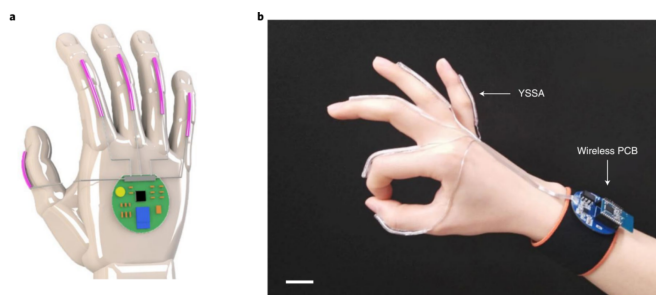


Figura 3.3: Rappresentazione schematica (a) e fotografia reale (b) della mano di un soggetto su cui è stato applicato il dispositivo YSSA, insieme a un trasmettitore wireless.[75]

levato costo di tali soluzioni, ad esempio il sistema "Talking Glove" [34] superava i 3.500 dollari, escludendo il costo del guanto stesso [15]. È opportuno evidenziare, tuttavia, che alcuni studi, come quelli presentati in [29, 52, 75] hanno cercato di affrontare questo problema mediante la progettazione di dispositivi a basso costo, riuscendo a contenere le spese entro i 50 dollari, con l'obiettivo di favorirne la diffusione e la commercializzazione. In secondo luogo, la natura stessa dei dispositivi indossabili introduce un fattore di intrusività che influisce negativamente sull'esperienza d'uso. Infine, la limitazione più profonda di questo approccio è di natura linguistica: i dispositivi non sono in grado di catturare le componenti non manuali (espressioni facciali, postura), che sono parti integranti e non accessorie della lingua dei segni. Questa semplificazione eccessiva della complessità linguistica è una delle cause principali del rifiuto di tali tecnologie da parte della comunità sorda.[15]

3.3.2. Approcci Vision-based

Le limitazioni intrinseche degli approcci basati su sensori, in termini di costi, intrusività e, soprattutto, incompletezza linguistica, hanno spinto la ricerca a concentrarsi su una seconda e oggi dominante metodologia: l'approccio vision-based. Utilizzando telecamere standard (es. RGB) o di profondità (es. RGB-D) e avanzati algoritmi di computer vision e deep learning, questo paradigma mira a riconoscere i segni analizzando direttamente il video del segnante. Questo metodo non solo elimina la necessità di dispositivi indossabili, ma offre anche il potenziale per catturare l'intera ricchezza espressiva della lingua dei segni, incluse le componenti non manuali.

Il primo lavoro significativo in questo campo risale al 1988, quando Tamura e Kawai esplorarono il riconoscimento di 20 parole chiave della Lingua dei Segni Giapponese (JSL), gettando le basi per le future ricerche in questo ambito [68].

Per l'elaborazione dell'informazione visiva, la letteratura scientifica ha storicamente

seguito due principali strategie, distinte in base al tipo di input fornito al modello di classificazione.

Approccio Frame Based

La prima è l'approccio frame-based [9, 16, 28, 36, 65], che utilizza direttamente le immagini del video (i frame) come input. In questo contesto, i modelli imparano a estrarre le caratteristiche salienti direttamente dai dati grezzi dei pixel. Questo approccio ha il grande vantaggio di poter catturare ogni dettaglio visivo, dalla texture della pelle alle deformazioni delle mani, fino alle espressioni facciali, senza alcuna perdita di informazione a priori. Tuttavia, l'elaborazione di interi fotogrammi risulta intrinsecamente onerosa dal punto di vista computazionale e richiede considerevoli risorse di calcolo. Le prestazioni dei modelli sono fortemente influenzate da fattori ambientali quali [50]:

- **Condizioni di illuminazione:** variazioni di luce, ombre nette o sovraesposizione possono alterare la percezione delle mani e del volto, rendendo il riconoscimento meno affidabile.
- **Background della scena:** uno sfondo complesso o dinamico può confondere gli algoritmi, che potrebbero faticare a isolare correttamente il segnante dal resto dell'immagine (un processo noto come *figure-ground segmentation*).
- **Occlusioni:** durante l'esecuzione dei segni, è comune che una mano nasconda l'altra, o che le mani ocludano parzialmente il volto o il corpo. Queste occlusioni comportano una perdita di informazione critica.
- **Variabilità del segnante e del punto di vista:** l'aspetto del segnante, il suo abbigliamento, la distanza e l'angolazione rispetto alla telecamera sono ulteriori variabili che il sistema deve essere in grado di gestire.

Approccio Skeleton Based

La seconda strategia, oggi particolarmente diffusa [27, 39, 45, 51, 72], è l'approccio skeleton-based, che si basa su una fase preliminare di pose estimation. Attraverso algoritmi specializzati, come OpenPose, MediaPipe [40] o modelli di *skeleton extraction*, ogni frame viene analizzato per individuare e tracciare un insieme di punti chiave (*keypoints*) sul corpo, sulle mani e sul volto del segnante. L'output non è un'immagine, ma una sequenza di coordinate spaziali (x, y, z) che descrive l'evoluzione della postura nel tempo. Questa rappresentazione, molto più compatta rispetto ai frame video, viene successivamente utilizzata come input per modelli progettati per elaborare dati strutturati come i grafi.

Questo secondo metodo risulta più robusto da influenze ambientali rispetto a quello frame based e la ridotta dimensione dei dati consente un'elaborazione computazionalmente più efficiente. Tuttavia, questa metodologia dipende fortemente dalla qualità dell'algoritmo di pose estimation: eventuali imprecisioni nel tracciamento dei keypoints si riflettono direttamente sulle prestazioni del modello di riconoscimento. Inoltre, dettagli fini come la configurazione esatta delle dita (quando non tracciata) o le informazioni testuali possono andare perse.

Approccio Ibrido

Entrambi gli approcci si sono dimostrati efficaci e non devono essere considerati mutuamente esclusivi. Al contrario, numerosi studi recenti propongono soluzioni ibride che integrano le informazioni visive, estratte dai frame RGB, con quelle cinematiche ricavate dai dati scheletrici, con l'obiettivo di sfruttare sinergicamente i punti di forza di entrambe le strategie. Non esiste un approccio univoco per combinare le informazioni visive con quelle derivate dall'estrazione dei punti chiave: ciascun lavoro di ricerca propone infatti una propria strategia di integrazione. Ad esempio, in [62] vengono utilizzate come input esclusivamente immagini RGB delle mani, dalle quali le articolazioni (*hand joints*) sono estratte tramite MediaPipe[40] e successivamente sovrapposte alle stesse immagini per arricchirne la rappresentazione visiva. Nel lavoro di Gan et al. [20], lo *skeleton* viene ricavato mediante una variante compressa della rete VGG seguita da due *deconvolution layers*, includendo non solo le mani ma anche il busto superiore. Questa rappresentazione scheletrica è quindi concatenata ai canali RGB e passata a una *Pseudo-3D Residual Network*. Inoltre, le *feature* ottenute dalla VGG vengono fuse con quelle generate dalla rete principale, al fine di costruire una rappresentazione multimodale più robusta e discriminativa, utile per la traduzione automatica della lingua dei segni. Un approccio differente è proposto in [74], dove le informazioni provenienti da RGB e *skeleton*, estratte tramite una rete HRNet pre-addestrata [66], vengono elaborate parallelamente e fuse soltanto nella fase finale. Infine, Jiang et al. [30] introducono un *framework* multimodale che combina canali RGB e profondità, elaborando ciascuna modalità separatamente e integrando le predizioni solo al termine della pipeline. Tale strategia consente di sfruttare la complementarità tra le diverse fonti di informazione, migliorando le prestazioni complessive del sistema di riconoscimento.

3.4. Modelli di Classificazione per l'ASLR

La scelta dell'architettura di classificazione in un sistema di ASLR è intrinsecamente legata alla natura e alla complessità della rappresentazione dei dati in ingresso. Le

metodologie spaziano da modelli di machine learning classico, adatti a dati a bassa dimensionalità, fino a complesse architetture di deep learning, necessarie per interpretare l'informazione proveniente da sorgenti video.

3.4.1. Classificazione di Dati da Sensori

Gli approcci sensor-based generano tipicamente dati sotto forma di serie temporali multivariate, dove ogni canale corrisponde alla lettura di un sensore (es. flessione di un dito, orientamento della mano). Data la natura strutturata e la dimensionalità relativamente contenuta di questi dati, è possibile impiegare con successo modelli di classificazione tradizionali. Gli approcci *sensor-based* generano tipicamente dati sotto forma di serie temporali multivariate, dove ogni canale corrisponde alla lettura di un sensore (es. flessione di un dito, orientamento della mano). Data la natura strutturata e la dimensionalità relativamente contenuta di questi dati, è possibile impiegare con successo modelli di classificazione tradizionali.

La letteratura riporta numerosi esempi di utilizzo di tecniche di *machine learning* classico nel riconoscimento della lingua dei segni. Ad esempio, in [75] viene proposto un approccio basato su una combinazione di più SVM (*multi-SVM*) per la classificazione dei segni a partire da dati raccolti tramite guanti sensorizzati. In [52], i dati di sensori a basso costo vengono analizzati con modelli quali KNN, Random Forest (RF) e SVM, mentre in [10] vengono confrontati SVM, KNN e reti neurali *feed-forward* (FNN) per la classificazione di segni isolati. Un lavoro più recente [29] estende l'analisi includendo Decision Tree (DT), SVM, KNN, RF e un modello *attention-based* BiLSTM, evidenziando come un modello più complesso che integri meccanismi di attenzione possa migliorare le prestazioni sui dati sequenziali. I vantaggi dell'utilizzo di un'architettura più complessa emergono anche in [24], dove viene proposto un modello ibrido *ConvNet + Attention*, capace di combinare l'estrazione automatica delle caratteristiche spaziali tramite convoluzioni con l'attenzione temporale, ottenendo una rappresentazione più ricca e informativa dei dati multimodali.

3.4.2. Classificazione di Dati Basati sulla Visione

La classificazione di dati vision-based è un problema intrinsecamente più complesso a causa dell'alta dimensionalità e della variabilità dell'input. In questo dominio, i modelli di deep learning sono diventati lo standard de-facto. La scelta dell'architettura dipende in modo cruciale dalla strategia di rappresentazione dei dati: frame-based o skeleton-based.

Approcci Frame-based

Gli approcci *frame-based* operano direttamente sulla sequenza di fotogrammi video, sfruttando modelli progettati per apprendere gerarchie di caratteristiche sia spaziali che temporali. A seconda della complessità dell'architettura, questi metodi spaziano da semplici reti convoluzionali 2D fino a modelli avanzati basati su Transformers, con un incremento progressivo della capacità di catturare informazioni multimodali e dipendenze di lungo raggio.

In uno dei primi studi che sfruttano reti convoluzionali per il riconoscimento della lingua dei segni [53], viene adottata una rete CNN 2D su singoli fotogrammi per classificare 20 segni della Lingua dei Segni Italiana (LIS), utilizzando dati in scala di grigi e informazioni di profondità (*depth*). Pur includendo anche CNN 3D per catturare la dimensione temporale, i risultati mostrano che, in presenza di dataset limitati, le CNN 2D ottengono una *validation accuracy* superiore, suggerendo che una modellazione temporale esplicita non garantisce necessariamente migliori prestazioni.

Per combinare informazioni spaziali e temporali in maniera più efficace, diversi studi hanno adottato architetture ibride CNN-LSTM. Ad esempio, [28] utilizza una ResNet per l'estrazione di caratteristiche da ciascun frame, seguita da una LSTM per modellare la dinamica temporale sui 64 segni del dataset LSA64. Un approccio ancora più sofisticato è presentato in [36], dove una CNN leggera basata su MobileNetV2 viene integrata con una LSTM dotata di meccanismo di attenzione, in modo da selezionare le informazioni più rilevanti all'interno delle sequenze video del dataset WLASL100. Infine, i modelli basati su Transformers hanno recentemente raggiunto prestazioni all'avanguardia. In particolare, [9] applica il Video Vision Transformer (ViViT) al riconoscimento di segni isolati appartenenti al dataset WLASL100, dimostrando un miglioramento significativo rispetto alle CNN tradizionali. Grazie al meccanismo di auto-attenzione, i Transformers sono infatti in grado di assegnare pesi differenziati a porzioni diverse di ciascun frame e a ciascun istante temporale, catturando dipendenze spaziali e temporali di lungo raggio con elevata efficacia.

Approcci Skeleton-based

Quando l'input è rappresentato come una sequenza temporale di coordinate dei key-point (scheletro), i modelli devono essere in grado di catturare sia la struttura cinematica sia la topologia del corpo umano. Nei primi lavori, le coordinate dei giunti venivano trattate come semplici vettori di caratteristiche, successivamente forniti a modelli classici di machine learning, come Support Vector Machines (SVM) o Artificial Neural Networks

(ANN) [32]. Sebbene efficaci in scenari limitati, tali approcci trascuravano le relazioni spaziali tra le articolazioni e la loro evoluzione temporale.

L'avvento delle *Graph Neural Networks* (GNN) ha segnato un punto di svolta, consentendo di modellare lo scheletro come un grafo in cui i nodi rappresentano le articolazioni e gli archi le connessioni anatomiche. Le *Graph Convolutional Networks* (GCN) e le loro estensioni spazio-temporali hanno permesso di apprendere in modo più fedele e robusto la dinamica del movimento e la struttura del corpo umano [46]. Ad esempio, [11] propone una *Skeleton-Based Graph Convolutional Network* (SL-GCN) che genera dati scheletrici da video e costruisce grafi temporali per catturare le relazioni spazio-temporali tra i giunti. Allo stesso modo, [45] estrae informazioni da più stream (joint, joint motion, bone e bone motion) applicando GCN dedicate a ciascun tipo di dato, mentre [51] introduce un meccanismo gerarchico di attenzione su grafi spazio-temporali con finestramento sia spaziale che temporale. L'integrazione di moduli spaziali dinamici e convoluzioni temporali multiscala viene proposta in [27], migliorando ulteriormente la capacità del modello di rappresentare relazioni complesse nel tempo.

Approcci Ibrido

I lavori che combinano informazioni scheletriche con dati RGB o di profondità combinano. Ad esempio, [30] adotta una strategia a flussi separati, in cui le previsioni provenienti da diverse modalità vengono generate indipendentemente e successivamente fuse: le 3D-CNN elaborano i dati RGB e di profondità, le GCN processano le informazioni scheletriche, mentre una rete *Separable Spatial-Temporal Convolutional* elabora le *features* dello scheletro. Analogamente, [74] impiega una D-ResNet per il canale RGB e una rete GCN multistream per lo scheletro, combinandone le predizioni tramite un ensemble finale. In [20], invece, le feature provenienti da GCN, VGG e reti convoluzionali-residuali sono fuse e processate con un *Transformer Encoder* e un *LSTM Decoder* per ottenere la sequenza di gloss. Infine, [62] propone un approccio semplificato in cui i keypoint delle mani vengono estratti tramite Mediapipe [40], da cui si ricavano feature relative alle distanze e agli angoli tra le articolazioni. Queste informazioni vengono poi combinate con feature ottenute da GoogleNet e infine elaborate da un classificatore SVM sulle caratteristiche fuse.

L'estrazione automatica delle coordinate avviene tipicamente tramite strumenti come Mediapipe[40], OpenPose o soluzioni personalizzate, ad esempio con *feature* derivate da VGG [20], o da reti come HRNet [66].

3.4.3. Modellazione delle Dipendenze Temporali nel Riconoscimento Continuo

Indipendentemente dalla modalità di acquisizione, affrontare il riconoscimento continuo richiede architetture capaci di modellare le dipendenze temporali e segmentare il flusso di input. In questo contesto, rivestono un ruolo centrale le Recurrent Neural Networks (RNN) e le Temporal Convolutional Networks (TCN) [2]. Le RNN, e le loro varianti più evolute come LSTM e GRU, sono state storicamente la scelta d'elezione per l'elaborazione di dati sequenziali. Più di recente, le TCN si sono affermate come una valida alternativa, utilizzando convoluzioni causali e dilatate per catturare pattern temporali su scale diverse, spesso con maggiore efficienza computazionale. A questi approcci di segmentazione implicita si affiancano strategie complementari orientate a una segmentazione esplicita dell'input. Un esempio è descritto in [52], dove la velocità delle mani viene usata come criterio euristico: un segno è considerato attivo solo quando la velocità supera una soglia. Questo permette di pre-segmentare il flusso di dati, semplificando il successivo compito di classificazione.

3.5. Metriche di Valutazione

La valutazione delle prestazioni di un sistema di ASLR rappresenta un passaggio fondamentale per determinarne l'efficacia e confrontarne i risultati con quelli di altri approcci presenti in letteratura. La scelta della metrica di valutazione più adeguata dipende strettamente dalla natura del compito affrontato: nel riconoscimento di segni isolati, assimilabile a un problema di classificazione multiclasse, sono preferibili metriche tradizionali come accuratezza, precision, recall e F1-score; al contrario, nel caso della traduzione di frasi continue, il problema si configura come una sequence-to-sequence, richiedendo metriche capaci di valutare l'intera predizione in termini sintattici e semantici; la metrica standard è il *Word Error Rate (WER)*, talvolta affiancata dalla metrica *BLEU*, particolarmente utile nei casi in cui il sistema operi una vera e propria traduzione automatica da lingua dei segni a lingua parlata o scritta.

3.6. Risultati e Criticità

Generalmente, le prestazioni riportate nei lavori sul riconoscimento automatico della lingua dei segni si attestano a un'accuratezza tra l'80% e il 99% nei casi in cui l'obiettivo è il riconoscimento di segni manuali isolati. Tali valori risultano invece leggermente inferiori, con un calo di alcuni punti percentuali, negli studi che affrontano il riconoscimento di segni

manuali continui e di componenti non manuali [42].

Tuttavia, tali risultati sono spesso influenzati da bias e non riflettono accuratamente la complessità dei contesti reali, in quanto ottenuti in condizioni sperimentali altamente controllate. Inoltre, molte delle valutazioni sono state condotte su un numero limitato di classi, come nel caso di [52], in cui sono state considerate solo 10 classi, o in [75], dove il sistema è stato testato su 11 classi.

Una problematica ricorrente nei progetti di *Automatic Sign Language Recognition* (ASLR) riguarda l'inadeguatezza dei dataset disponibili. Negli approcci *sensor-based*, la creazione e la diffusione di dataset pubblici risultano fortemente limitate dalle specificità del sistema utilizzato per la raccolta dei dati. Di conseguenza, molti studi si basano su un numero ridotto di classi, come avviene in [52], dove sono stati considerati soltanto 10 segni, o in [24], in cui sono state analizzate 26 lettere dell'alfabeto ASL.

Gli approcci *vision-based* presentano invece limitazioni sia di natura qualitativa che quantitativa. Dal punto di vista qualitativo, molti dataset disponibili sono costituiti da registrazioni realizzate da persone comuni che, non essendo segnanti professionisti, possono eseguire i segni in modo impreciso o non pienamente conforme agli standard linguistici. Dal punto di vista quantitativo, invece, la scarsità di dati è dovuta, da un lato, all'assenza di dataset di grandi dimensioni condivisi a livello internazionale e, dall'altro, alla natura intrinsecamente dialettale delle lingue dei segni e al loro riconoscimento ufficiale relativamente recente in numerosi Paesi.

Negli ultimi anni sono stati proposti diversi dataset che hanno parzialmente colmato questa lacuna. Tra i più utilizzati in ambito accademico vi è il *Word-Level American Sign Language* (WLASL) [38], che contiene oltre 2.000 segni realizzati da più di 100 segnanti, sebbene nella pratica venga spesso adottata una versione ridotta con soltanto 100 segni [9, 36]. Altri dataset di riferimento includono MS-ASL [31], con 25.000 video relativi a 1.000 segni, SIGNUM [71], che fornisce un corpus per la Lingua dei Segni Tedesca con 450 segni e 780 frasi per oltre 30.000 video, e RWTH-PHOENIX-Weather 2014 [19], che contribuisce con oltre 75.000 video di lingua dei segni continua, completi di annotazioni della posa delle mani.

In molti studi non incentrati sulla ASL il numero di classi considerate è spesso limitato. Ad esempio, in [62] vengono utilizzate esclusivamente le 41 lettere dell'alfabeto della Lingua dei Segni Giapponese (JSL). Il dataset LSA64 [56], invece, fornisce 3.200 video relativi a 64 segni della Lingua dei Segni Argentina, risultando più ricco per quantità di segni ma comunque ridotto rispetto agli standard dei moderni modelli di *Deep Learning*.

Anche la Lingua dei Segni Italiana (LIS) presenta una significativa scarsità di dataset pubblici. Come evidenziato in [11, 47], i dataset di dimensioni più consistenti sono spesso di natura privata. Tra le principali risorse disponibili si annoverano *SpreadTheSign*[64][25], progetto avviato nel 2004 come dizionario online multilingue per diverse lingue dei segni, e il *Corpus LIS*[12], considerata la raccolta più ampia di video in LIS, comprendente contenuti spontanei, semi-strutturati e strutturati prodotti da persone sorde.

Nonostante la ricchezza in termini di varietà di segni, queste risorse non risultano adatte all'addestramento di modelli di classificazione automatica, principalmente a causa del numero limitato di video disponibili per ciascun segno. Ad oggi, il dataset pubblico più rilevante per la LIS è *MEDALIS* [47], incentrato sui segni utilizzati in ambito medico: si tratta di un dataset di segni isolati composto da 126 classi e oltre 12.000 registrazioni, comprendenti sia video RGB che dati Radar Doppler (RDM) acquisiti tramite un sistema radar a 60 GHz. Un'ulteriore risorsa è *A3LIS-147* [17], sviluppato da un gruppo di ricercatori dell'Università Politecnica delle Marche, anch'esso costituito da segni isolati ma con un numero sensibilmente inferiore di campioni, pari a soli 10 video per segno, risultando quindi meno adatto per l'addestramento di modelli di *Deep Learning* su larga scala.

Il presente lavoro si propone di fornire un'analisi comparativa dei principali metodi di classificazione della Lingua dei Segni Italiana (LIS), con un focus specifico su approcci di *Deep Learning*. In particolare, viene adottato un approccio *vision-based*, e più precisamente *skeleton-based*: tramite MediaPipe[40], da ciascun frame vengono estratti una serie di *keypoints* da cui vengono calcolati 42 angoli, tenendo conto non solo delle componenti manuali, ma anche di quelle non manuali.

L'analisi è condotta su due *dataset* privati forniti dall'azienda Cludia Research e contenenti lettere, numeri e parole di uso quotidiano: il primo comprende 46 segni, il secondo 72, per un totale di 9.700 video, con oltre 100 campioni per ciascun segno nel *dataset* più esteso. È importante sottolineare che non è stato possibile utilizzare *dataset* pubblici disponibili online, a causa di limitazioni significative quali: numero ridotto di video, dimensioni dei *frame* inferiori agli standard richiesti ed elevata variabilità dei segni associati alla stessa classe. Nei *dataset* impiegati in questo lavoro, invece, ogni classe è associata in modo univoco a un singolo segno, garantendo maggiore coerenza e qualità dei dati per l'addestramento dei modelli.

Il contributo innovativo di questa tesi si articola su tre compiti principali:

- **Classificazione di segni isolati:** sono stati impiegati modelli di *Machine Learning* classico come baseline, ma soprattutto architetture di tipo convoluzionale (CNN),

Long Short-Term Memory (LSTM) e *Transformers*, al fine di confrontare le diverse capacità di modellazione temporale e spaziale dei dati.

- **Out-of-Distribution (OOD) Detection:** l'obiettivo è identificare se un segno appartenga o meno alla distribuzione vista in fase di addestramento. Sono stati analizzati diversi approcci, tra cui metodi di *anomaly detection* sugli *embedding* della rete convoluzionale addestrate, tecniche basate sull'analisi dei *logits* (valore massimo, entropia, energia) confrontati con soglie predefinite, e un approccio *teacher-student*.
- **Classificazione di segni continui:** oltre a una baseline che sfrutta la rete addestrata sui segni isolati, sono stati valutati due approcci principali: uno basato su *sliding window* con *majority voting*, che considera finestre temporali limitate, e un approccio *end-to-end* mediante una rete LSTM.

Questo studio fornisce quindi un'analisi comparativa approfondita delle prestazioni ottenute nei diversi scenari, offrendo una valutazione sia quantitativa che qualitativa dei metodi proposti.

Il presente lavoro si configura come uno dei primi studi nel contesto italiano a utilizzare un *dataset* con oltre 70 classi univoche e più di 100 campioni per segno, incentrato su parole di uso quotidiano piuttosto che su domini specifici. L'approccio proposto integra, per la prima volta nell'ambito del riconoscimento di segni appartenenti alla LIS, un'analisi congiunta di segni isolati e continui basata su *skeleton-based features* e reti neurali, estendendo inoltre la ricerca alla *Out-of-Distribution Detection*, tema finora poco esplorato nel riconoscimento automatico della lingua dei segni.

4 | Metodi

La traduzione automatica della lingua dei segni rappresenta una sfida particolarmente complessa nel campo dell'intelligenza artificiale. Sebbene siano stati compiuti notevoli progressi nei settori del Deep Learning e della computer vision, lo sviluppo di sistemi automatici in grado di interpretare in modo accurato e fluido la comunicazione segnica è ancora in fase esplorativa. In particolare, negli ultimi anni si è assistito a un progressivo spostamento dagli approcci basati su dispositivi indossabili, come guanti sensorizzati o accelerometri, verso soluzioni che sfruttano esclusivamente informazioni visive estratte da sequenze video. Questo cambio di paradigma, reso possibile dall'evoluzione dei modelli di apprendimento automatico, apre nuove prospettive per la creazione di sistemi non invasivi e accessibili, basati su strumenti comunemente disponibili.

Il presente capitolo descrive in dettaglio i metodi sviluppati e analizzati nel corso di questo lavoro. Dopo aver illustrato le tecniche di costruzione e preprocessing del dataset, verranno presentate le architetture di rete implementate per la classificazione dei segni isolati, le strategie per la gestione di sequenze contenenti più segni e gli approcci proposti per il rilevamento di segni non appartenenti alla distribuzione di addestramento (*out-of-distribution detection*).

4.1. Preparazione del Dataset

Nel contesto di questo progetto, sono stati raccolti oltre 10.000 video grazie alla collaborazione con l'azienda Cludia Research e con il supporto di un gruppo di segnanti madrelingua o interpreti LIS qualificati. Si tratta, ad oggi, di uno dei primi tentativi su larga scala di acquisizione sistematica di dati visivi in Lingua dei Segni Italiana (LIS). Ogni video documenta un'espressione singola in LIS ed è accompagnato dalla relativa trascrizione testuale in lingua italiana, rendendo possibile l'utilizzo di tali dati per attività di apprendimento supervisionato.

I video raccolti hanno una risoluzione di 1080×720 pixel, sono codificati nel formato WebP, mantengono una frequenza costante di 30 frame al secondo e hanno una durata

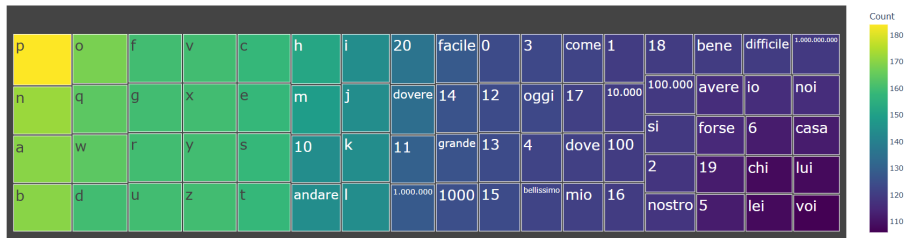


Figura 4.1: Classi che compongono il dataset di 72 segni con i relative frequenze

complessiva di 5 secondi. I modelli sviluppati in questo progetto non operano direttamente sulle immagini raw, ma si avvalgono di un processo di preprocessing basato su MediaPipe[40], una libreria che consente l'estrazione di landmark anatomici significativi per ogni frame. Da questi landmark vengono calcolati 42 angoli relativi alla posizione di dita, mani e bocca.

I valori angolari, inizialmente compresi nell'intervallo $[-180^\circ, 180^\circ]$, sono stati normalizzati su un intervallo standardizzato da -1 a 1 per favorire la stabilità numerica e l'efficacia dell'addestramento dei modelli.

La costruzione del dataset è avvenuta in maniera incrementale: si è partiti da una versione iniziale con soli 6 segni, utilizzata come *proof of concept*, per poi estendersi a 26 segni corrispondenti alle lettere dell'alfabeto. Successivamente, è stata definita una composizione intermedia di 46 segni, che ha permesso di condurre una prima fase di sperimentazioni più estesa, fino ad arrivare all'attuale versione di 72 segni, comprendente l'intero alfabeto e alcune parole di uso frequente. Questi ultimi due dataset, rispettivamente a 46 e 72 segni, costituiscono le basi principali delle analisi e dei confronti riportati in questo lavoro.

Durante lo sviluppo del *dataset* sono stati riscontrati diversi problemi, in particolare legati alla natura non standardizzata dei video, derivanti dall'uso di strumentazioni differenti per la registrazione. Alcuni video contenevano artefatti visivi o rumore di fondo, per esempio il transito di soggetti non pertinenti. Per mitigare tali problemi, si è implementato un metodo basato su una griglia spaziale per identificare con precisione l'inizio effettivo del segno all'interno del video, consentendo così di isolare e utilizzare solo la porzione rilevante del filmato. Tuttavia, anche dopo questa fase di pulizia, si sono osservate leggere variazioni nel numero di frame tra le diverse sequenze, dovute a differenze nei movimenti o a piccole discrepanze temporali. Per garantire una rappresentazione omogenea e confrontabile tra tutte le istanze, è stato quindi necessario uniformare la lunghezza delle sequenze estratte: a quelle aventi un numero di frame inferiore allo standard è stato applicato un post-padding con valore nullo per uniformare la lunghezza delle sequenze.

Per l'addestramento e la valutazione dei modelli, il dataset è stato suddiviso in tre sottoinsiemi disgiunti: **training** (80%), **validation** (10%) e **test** (10%). Questa suddivisione è stata effettuata in modo stratificato rispetto alle classi, al fine di preservare una distribuzione equilibrata dei segni all'interno di ciascun sottoinsieme.

4.2. Classificazione di un singolo Segno

In questa sezione viene descritta la fase iniziale del progetto, dedicata alla classificazione di segni isolati. Partendo dal *dataset* precedentemente descritto, l'obiettivo principale è stato quello di progettare e valutare modelli in grado di classificare correttamente la classe di appartenenza di ciascun segno, a partire da una sequenza di 42 *features* estratte. Tali *features*, ottenute mediante una fase di pre-processing dedicata, costituiscono l'input per le diverse architetture di rete analizzate nelle sezioni successive.

4.2.1. Individuazione delle Baseline

Inizialmente, è stata condotta un'analisi esplorativa impiegando algoritmi di Machine Learning tradizionali. Lo scopo di questa fase era duplice: stabilire una baseline prestazionale di riferimento e valutare l'efficacia di approcci non basati su reti neurali. A tal fine, sono stati testati diversi modelli, dai più semplici come la Regressione Logistica e le Support Vector Machine (SVM), a metodi basati su alberi decisionali come Decision Tree e Random Forest. Si precisa che non è stata condotta un'ottimizzazione sistematica degli iperparametri (hyperparameter tuning), poiché l'obiettivo era unicamente ottenere una stima rapida delle performance raggiungibili con tali metodi.

Successivamente, l'analisi si è concentrata sull'impiego di reti neurali. È stato sviluppato un primo modello di riferimento, basato su un Multi-Layer Perceptron (MLP). Questa rete era composta da tre strati densi (*fully-connected*), a cui sono stati integrati meccanismi di regolarizzazione come il Dropout e la Batch Normalization per migliorare la capacità di generalizzazione e mitigare il rischio di overfitting. Questo approccio basilare ha permesso di verificare la capacità delle reti neurali, anche in una configurazione minimale, di apprendere rappresentazioni significative (feature learning) a partire dai dati di input, ponendo le basi per lo sviluppo di architetture più complesse.

4.2.2. Sviluppo di Architetture di Deep Learning Avanzate

Per migliorare le performance ottenute con i modelli di riferimento, sono state esplorate tre famiglie di architetture di Deep Learning:

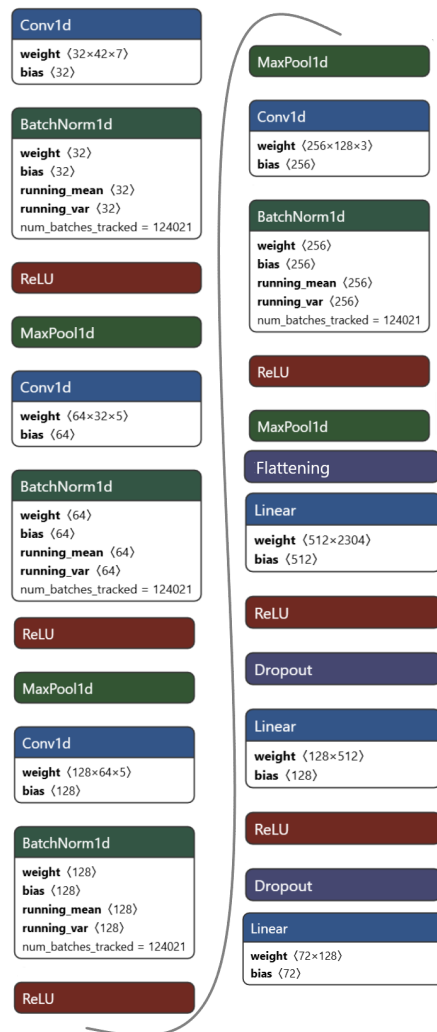


Figura 4.2: Architettura di una rete neurale con CNN monodimensionale

Reti basate su CNN monodimensionali (1D-CNN) (fig. 4.2): Le reti 1D-CNN sono state scelte per la loro capacità di catturare **pattern locali** lungo la dimensione temporale in modo computazionalmente efficiente, sfruttando operazioni convoluzionali che possono essere eseguite in parallelo. A differenza delle reti ricorrenti, le CNN non modellano in maniera esplicita le dipendenze temporali a lungo termine, ma risultano particolarmente efficaci nell'estrazione di caratteristiche locali e ripetitive, che costituiscono informazioni cruciali nei segnali temporali come quelli della lingua dei segni.

Nella sezione iniziale della rete sono stati inseriti una serie di blocchi convoluzionali il cui obiettivo principale è l'estrazione automatica delle caratteristiche più informative dalla sequenza di input. Ciascun blocco è progettato per trasformare progressivamente i dati grezzi in rappresentazioni di livello sempre più astratto, facilitando così la fase di

classificazione finale. Ogni blocco è composto da:

- una **convoluzione 1D (Conv1D)** con *kernel size* variabile, per permettere al modello di catturare dipendenze a diverse scale temporali (kernel piccoli per pattern locali, kernel più ampi per dipendenze di ordine superiore);
- una **Batch Normalization**, per stabilizzare e accelerare la convergenza dell'addestramento;
- una funzione di attivazione **ReLU**, che introduce non-linearità e migliora la capacità espressiva del modello;
- una **Max Pooling** per ridurre la dimensionalità e aumentare l'invarianza alle piccole traslazioni temporali.

In seguito all'estrazione delle *features* tramite i blocchi convoluzionali, l'output dell'ultimo blocco viene appiattito (*flattened*) e fornito in input a uno o più strati fully-connected. La dimensionalità di tali strati rappresenta un ulteriore iperparametro chiave, ottimizzato durante la fase di ricerca per bilanciare capacità espressiva del modello e rischio di overfitting.

Per migliorare la capacità di generalizzazione, tra gli strati densi sono stati inseriti strati di Dropout, con probabilità di disattivazione selezionate anch'esse tramite ottimizzazione iperparametrica.

La rete si conclude con uno strato di classificazione dotato di funzione di attivazione Softmax, incaricato di produrre la distribuzione di probabilità sulle diverse classi di segni.

L'esplorazione dello spazio degli iperparametri ha permesso di identificare la configurazione ottimale, riportata in figura 4.2, composta da 4 blocchi convoluzionali con *kernel size* rispettivamente pari a 7, 5, 3 e 3, seguiti da 2 strati fully-connected di dimensione 512 e 128 unità e da strati di Dropout con probabilità pari a 0.4 e 0.3.

Reti basate su LSTM (fig 4.3):

Poiché la lingua dei segni è intrinsecamente sequenziale, è fondamentale disporre di modelli in grado di cogliere le relazioni temporali tra i frame. A questo scopo sono state utilizzate le Long Short-Term Memory (LSTM), una tipologia di reti ricorrenti capace di apprendere dipendenze anche di lungo periodo.

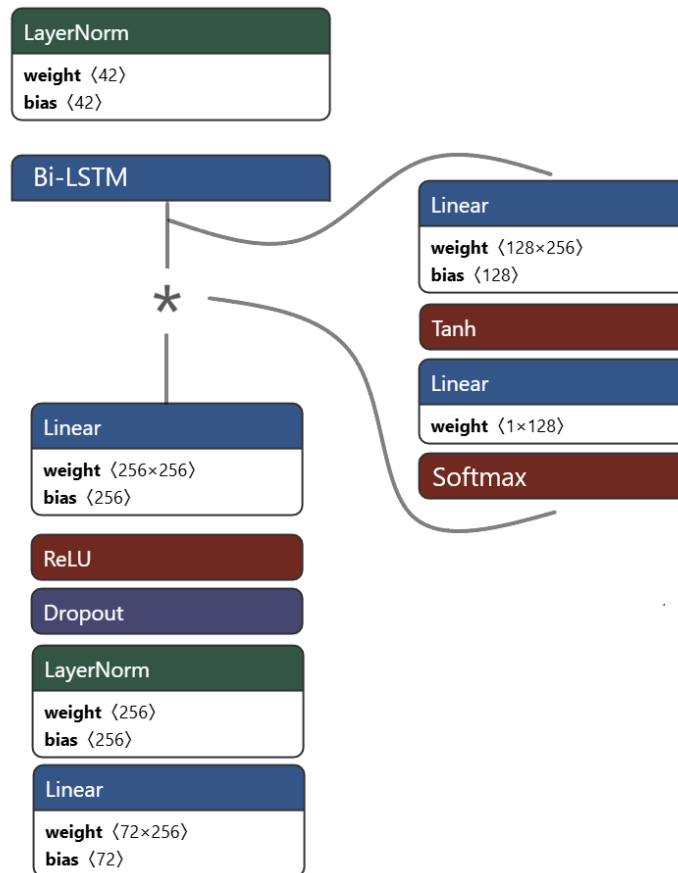


Figura 4.3: Architettura di una rete neurale basata su Bi-LSTM e meccanismo di attenzione

Sono state considerate sia configurazioni unidirezionali che bidirezionali (Bidirectional LSTM). In quest'ultimo caso, la rete è costituita da due LSTM parallele che processano la sequenza rispettivamente in ordine diretto e inverso, consentendo di integrare informazioni provenienti sia dal passato che dal futuro rispetto a ogni punto della sequenza, migliorando così la capacità di catturare pattern complessi.

Per l'aggregazione dell'informazione temporale e la classificazione finale sono state sperimentate diverse strategie:

- **Classificazione basata sull'ultimo stato nascosto:** utilizza l'output dell'ultimo *time step* come rappresentazione compatta dell'intera sequenza, successivamente fornita a uno strato di classificazione.
- **Classificazione a maggioranza (Majority Voting):** applica un classificatore a ciascun output temporale della LSTM e combina le predizioni tramite un voto a maggioranza per determinare la classe finale.

- **Meccanismo di Attenzione:** introduce uno strato di attenzione che apprende a pesare dinamicamente i contributi dei diversi *time step*, permettendo al modello di focalizzarsi sui frame più informativi. Il vettore di contesto così ottenuto viene poi utilizzato per la predizione finale.

La struttura generale delle diverse architetture esplorate, illustrata in figura 4.3, segue una sequenza di moduli distinti, ognuno con una funzione specifica:

- **Pre-elaborazione dell'input:** la sequenza temporale viene inizialmente sottoposta a un blocco di normalizzazione, applicato a ciascun *time step*, con l'obiettivo di stabilizzare la distribuzione dei dati e agevolare la convergenza del modello.
- **Blocco Bi-LSTM:** successivamente, i dati vengono processati da una rete LSTM bidirezionale caratterizzata da un numero variabile di livelli ricorrenti (*hidden layers*) e da una specifica *hidden dimension*. Questa componente permette di catturare dipendenze temporali sia passate che future all'interno della sequenza.
- **Meccanismo di attenzione:** l'output del blocco Bi-LSTM viene quindi fornito a un modulo di attenzione, costituito da più *linear layers* intervallati da una funzione di attivazione *tanh*, con una *softmax* finale che normalizza l'output. Tale meccanismo apprende a pesare dinamicamente i contributi temporali, moltiplicando i pesi di attenzione per l'output della Bi-LSTM, in modo da enfatizzare i frame più rilevanti per la classificazione finale.
- **Sezione fully-connected:** infine, il vettore risultante viene passato a una serie di strati densi, composti da una *linear layer*, una funzione di attivazione ReLU, uno strato di *dropout* e un'operazione di *normalizzazione*, per poi concludersi con uno strato di classificazione dotato di funzione *softmax*.

La configurazione più efficace utilizza una *hidden dimension* pari a 128 per i blocchi ricorrenti, seguiti da un meccanismo di attenzione costituito da due strati lineari con rispettivamente 256 e 128 unità. L'output pesato viene quindi elaborato da un ulteriore strato *fully-connected* di 256 unità, con un tasso di *dropout* pari a 0.3 per mitigare il rischio di overfitting, prima della fase di classificazione finale.

Reti basate su Transformer (fig 4.4): Infine, è stato considerato un approccio più recente e flessibile, basato sull'architettura Transformer, in grado di apprendere dipendenze di lungo periodo all'interno delle sequenze senza ricorrere a meccanismi ricorrenti, ma sfruttando meccanismi di attenzione che consentono di modellare in modo esplicito le relazioni tra tutti gli elementi della sequenza.

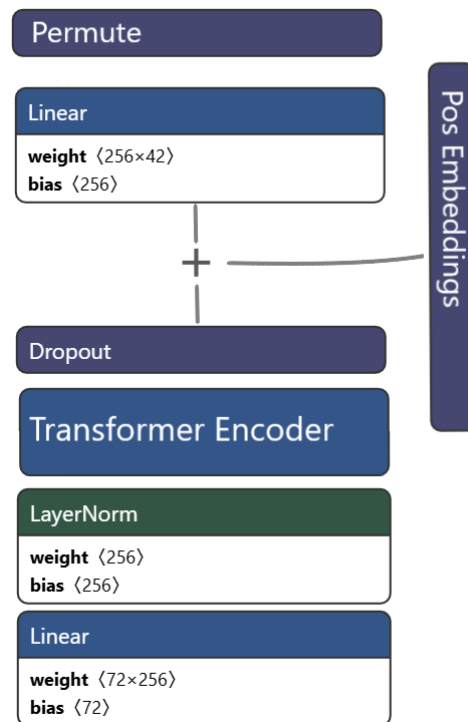


Figura 4.4: Architettura di un modello basato su Transformer Encoder

Il blocco fondamentale di queste architetture è rappresentato dal *Transformer Encoder*, che attraverso il meccanismo di *self-attention* permette a ciascun elemento della sequenza di interagire con tutti gli altri, pesando dinamicamente il contributo informativo di ogni posizione in funzione del contesto globale.

Nelle architetture proposte, di cui un esempio è illustrato in figura 4.4, l'input viene innanzitutto proiettato in uno spazio latente di dimensione D , a cui viene concatenato un token speciale [CLS] utilizzato per la classificazione. Per preservare l'informazione temporale, alla sequenza viene poi aggiunto un positional embedding, sperimentato sia nella forma fissa sia in quella *learnable*.

La sequenza così ottenuta viene quindi elaborata da uno o più *Transformer Encoder*, composti da moduli di *multi-head self-attention* e da *feed-forward network* posizionali, ciascuno seguito da operazioni di *residual connection* e *layer normalization*.

Infine, l'output associato al token [CLS] viene passato a uno o più strati fully-connected, con funzione di attivazione *Softmax* nello strato finale, per ottenere la distribuzione di probabilità sulle classi considerate.

L'architettura ottimale individuata proietta l'input in uno spazio latente di dimensione 256, adotta un tasso di dropout pari a 0.2 per ridurre il rischio di overfitting e impiega 8

attention heads all'interno di 3 *hidden layers*, garantendo così un efficace bilanciamento tra capacità espressiva e complessità computazionale.

4.2.3. Utilizzo delle Funzioni di Loss

La scelta della *loss function* è un elemento cruciale che influenza direttamente la convergenza del modello e la qualità delle rappresentazioni latenti apprese. Per ottimizzare i modelli sviluppati in questo lavoro, sono state esplorate e confrontate diverse strategie, che vanno oltre la tradizionale Cross-Entropy categorica per includere approcci avanzati basati sull'apprendimento metrico (*metric learning*). Inoltre, è stata valutata l'efficacia di approcci ibridi, ottenuti dalla combinazione lineare di diverse funzioni di perdita.

La **Categorical Cross-Entropy** rappresenta la scelta canonica per problemi di classificazione multiclasse. Essa misura la distanza tra la distribuzione predetta dal modello e la distribuzione reale (*one-hot encoding*) e viene minimizzata quando il modello assegna probabilità massima alla classe corretta. Per gestire il lieve sbilanciamento del dataset, dovuto alla diversa numerosità delle classi, sono stati sperimentati anche approcci di *weighted loss*, attribuendo a ciascuna classe un peso inversamente proporzionale alla sua frequenza.

$$\mathcal{L}_{\text{WCE}} = - \sum_{i=1}^N \sum_{c=1}^C w_c \cdot y_{i,c} \cdot \log(\hat{y}_{i,c}) \quad (4.1)$$

Accanto a questa soluzione standard, sono stati esplorati approcci più avanzati, tipici del paradigma dell'apprendimento *metric-based*, in particolare nell'ambito delle *Siamese Networks*. In tale contesto è stata adottata la **Contrastive Loss**, la quale riceve in input due campioni e una label binaria indicante se i due esempi appartengono alla stessa classe. La funzione agisce sugli *embedding* ottenuti dalla penultima layer del modello (prima dello strato di classificazione finale) e mira a minimizzare la distanza tra campioni simili e massimizzare la distanza tra quelli di classi differenti, entro un margine predefinito. Questo tipo di formulazione incoraggia il modello a costruire uno spazio latente più strutturato e semantico, nel quale le classi siano meglio separabili.

$$\mathcal{L}_{\text{contrastive}} = \frac{1}{2N} \sum_{i=1}^N (y_i \cdot D_i^2 + (1 - y_i) \cdot \max(0, m - D_i)^2) \quad (4.2)$$

Infine, è stata impiegata anche la **Triplet Loss**, che estende il concetto precedente considerando triplette di campioni: un'*anchor*, un esempio *positivo* appartenente alla

stessa classe, e un *negativo* di classe differente. La funzione di loss incentiva il modello a ridurre la distanza tra l'anchor e il positivo, e contemporaneamente ad aumentare la distanza tra l'anchor e il negativo, garantendo che quest'ultima sia almeno superiore a una soglia marginaria M . Questo meccanismo ha dimostrato particolare efficacia nella costruzione di rappresentazioni robuste in scenari con elevata variabilità intra-classe.

$$\mathcal{L}_{\text{triplet}} = \frac{1}{N} \sum_{i=1}^N [\|f(x_i^a) - f(x_i^p)\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + M]_+ \quad (4.3)$$

In fase sperimentale, sono state inoltre testate **combinazioni lineari** tra la *Categorical Cross-Entropy* e una delle loss metriche (Contrastive o Triplet), mediante un coefficiente di bilanciamento $\alpha \in [0, 1]$. Tale approccio si è rivelato utile per integrare le capacità di classificazione supervisionata con quelle di apprendimento di uno spazio semantico strutturato.

$$\mathcal{L}_{\text{combinata}} = \alpha \cdot L_{\text{WCE}} + (1 - \alpha) \cdot L_{\text{contrastive/triplet}} \quad (4.4)$$

4.2.4. Strategie di Addestramento e Ottimizzazione degli Iperparametri

Per garantire un confronto equo tra i diversi modelli, tutti gli esperimenti sono stati eseguiti per un numero massimo di 300 epoche. Per prevenire l'overfitting e migliorare la capacità di generalizzazione, è stato sistematicamente impiegato un meccanismo di Early Stopping. Tale meccanismo monitora la loss riferita al set di validazione e interrompe il processo di addestramento qualora non si registrino miglioramenti per un numero consecutivo di epoche, definito come *patience*; negli esperimenti condotti, questo parametro ha assunto un valore compreso tra le 10 e 20 epoche. Come punto di partenza per l'ottimizzazione, è stato utilizzato l'ottimizzatore Adam, data la sua comprovata efficacia in un'ampia gamma di applicazioni di Deep Learning, e un *batch size* di 32. Tuttavia, è fondamentale sottolineare che questi valori non sono stati fissati a priori, ma sono stati considerati essi stessi iperparametri da ottimizzare.

La gestione del learning rate è stata un elemento centrale della strategia di addestramento. Sebbene sia stato definito un valore di partenza per la fase di tuning (es. 10^{-3}), sono state adottate strategie di scheduling dinamico per adattarlo durante l'addestramento. In particolare, sono stati sperimentati due scheduler:

- **ReduceLROnPlateau**: riduce il tasso di apprendimento di un fattore predefinito (es. 0.1) ogni qualvolta la metrica di validazione entra in una fase di stallo (plateau).
- **Cosine Annealing Scheduler**: adatta il learning rate seguendo un andamento cosinusoidale per ogni ciclo di addestramento, diminuendolo gradualmente da un valore massimo a un minimo. Questa tecnica si è dimostrata efficace nel favorire l'esplorazione di minimi locali più ampi e robusti.

Infine, per identificare la configurazione ottimale di ciascuna architettura, è stata condotta una fase di ottimizzazione automatica degli iperparametri (HPO). Anziché affidarsi a una ricerca manuale o a una griglia esaustiva (Grid Search), sono stati utilizzati framework specializzati come Optuna, che implementano algoritmi di ricerca bayesiana. Questo ha permesso di esplorare in modo efficiente un vasto spazio di ricerca, che includeva iperparametri chiave quali:

- Il numero di blocchi convoluzionali o ricorrenti.
- Il valore del learning rate (tra 10^{-2} e 10^{-5}).
- La dimensione degli strati nascosti e dello spazio di embedding.
- Il valore della probabilità di Dropout (tra 0.1 e 0.5).
- L'ottimizzatore da utilizzare (es. Adam, SGD) e i relativi parametri (es. beta, momentum).
- La dimensione del batch (es. 16, 32, 64).
- La presenza e la posizione degli strati di Batch Normalization.
- L'inclusione di connessioni residue per facilitare il flusso del gradiente.

Questo approccio sistematico ha garantito che ogni modello fosse valutato nella sua configurazione più performante.

4.3. Out Of Distribution Detection

Questa sezione affronta il problema del riconoscimento in un contesto open-set, in cui il sistema può incontrare campioni non appartenenti a nessuna delle classi note (out-of-distribution).

L'introduzione di questo componente si è resa necessaria sia per il numero limitato di segni attualmente supportati dal sistema di riconoscimento, sia per la natura stessa della Lingua dei Segni Italiana, che si distingue per un'elevata varietà e flessibilità espressiva.

In un contesto reale, infatti, è plausibile che il sistema venga esposto a segni non presenti nel set di addestramento.

Per affrontare questo scenario, è stato progettato un meccanismo in grado di rilevare in maniera esplicita i segni "sconosciuti", evitando che vengano erroneamente classificati come appartenenti a una delle classi conosciute. Questo approccio contribuisce a migliorare la robustezza e l'affidabilità del sistema, consentendo di segnalare un'istanza come segno non riconosciuto, piuttosto che forzare una classificazione potenzialmente fuorviante.

Per condurre questa analisi, è stato adottato un approccio supervisionato parzialmente controllato: sono state selezionate casualmente 40 classi del dataset come dati normali e altre 6 classi sono state considerate anomalie durante la fase di valutazione. Questo ha permesso di simulare un contesto realistico, in cui il sistema è esposto anche a segni non presenti nel set di addestramento.

4.3.1. Approccio tramite Embedding con Anomaly Detection

In una prima fase, è stata progettata una pipeline in due stadi. Gli input vengono inizialmente trasformati in *embedding* mediante una rete neurale convoluzionale monodimensionale (1D CNN); la rete è stata addestrata con funzioni obiettivo specifiche per l'apprendimento di embedding discriminativi, quali la *Contrastive Loss* e la *Triplet Loss*. Entrambe incentivano la vicinanza nello spazio latente tra istanze appartenenti alla stessa classe e aumentano la distanza tra quelle di classi differenti, promuovendo così una maggiore separabilità tra le distribuzioni.

Gli *embedding* così ottenuti sono stati successivamente analizzati tramite algoritmi classici di *outlier detection*, con l'obiettivo di discriminare tra esempi appartenenti al dominio di addestramento e potenziali istanze OOD. Tra i metodi considerati figurano:

- **K-means**, che assegna i punti agli k cluster più vicini, considerando le istanze distanti dai centroidi come potenziali anomalie;
- **DBSCAN**, che identifica come rumore i punti non assegnabili a cluster densi;
- **Isolation Forest**, che si basa sul principio per cui gli outlier risultano più facilmente isolabili tramite partizionamenti ricorsivi dello spazio dei dati.

Come verrà mostrato nella Sezione 5.2.1, i risultati ottenuti evidenziano i limiti intrinseci di tali approcci, fornendo al contempo la motivazione per l'adozione di metodi più avanzati.

4.3.2. Approccio basato sull'analisi dei logit

Un ulteriore approccio, strettamente collegato al modello di classificazione, si basa sull'analisi dei logit generati in uscita da una rete neurale addestrata per il riconoscimento dei singoli segni, descritta nel Capitolo Single Sign Classification. In questo caso, la rete utilizzata per l'Out of Distribution Detection, in termini di architettura, coincide esattamente con quella impiegata per la classificazione. L'addestramento del modello è stato effettuato utilizzando una loss function combinata, che include la Cross Entropy Loss e una tra la Contrastive Loss o la Triplet Loss, al fine di rafforzare la struttura dello spazio degli embedding e aumentare la separabilità tra classi differenti. Questo schema di apprendimento multi-obiettivo consente alla rete non solo di classificare correttamente i segni, ma anche di apprendere rappresentazioni interne utili per distinguere segni appartenenti al dominio noto da quelli anomali.

L'analisi dei logit prodotti dalla rete consente di implementare diverse strategie di *anomaly detection*, basate su metriche che riflettono l'incertezza del modello rispetto all'istanza in input. In particolare, sono state considerate le seguenti tre misure:

- Il valore massimo della distribuzione predetta
- L'entropia della distribuzione dei logit softmax-scalati
- L'energia del vettore dei logit

Ciascuna di queste metriche viene calcolata a partire dai logit $\mathbf{z} = (z_1, z_2, \dots, z_C) \in \mathbb{R}^C$, dove C è il numero delle classi. I logit vengono opportunamente scalati tramite un parametro di temperatura $T > 0$, ottenendo:

$$\mathbf{p}_T = \text{softmax}\left(\frac{\mathbf{z}}{T}\right), \quad \text{con } p_i = \frac{\exp(z_i/T)}{\sum_{j=1}^C \exp(z_j/T)}$$

Le metriche utilizzate sono definite come segue:

Valore massimo della probabilità (confidence massima):

$$\text{MaxProbability} = \max_i (p_i)$$

Entropia della distribuzione predetta:

$$\mathcal{H}(\mathbf{p}_T) = - \sum_{i=1}^C p_i \cdot \log(p_i + \varepsilon)$$

dove ε è un termine di stabilizzazione numerica per evitare logaritmi nulli.

Energia del vettore dei logit (Energy Score):

$$\mathcal{E}(\mathbf{z}) = -T \cdot \log \left(\sum_{i=1}^C \exp \left(\frac{z_i}{T} \right) \right)$$

Infine, ciascuna di queste metriche viene confrontata con una soglia predefinita, scelta empiricamente o ottimizzata su un insieme di validazione. Il confronto con la soglia permette di stabilire se l'istanza in esame debba essere considerata come un outlier, e quindi classificata come *segno non riconosciuto*.

4.3.3. Approccio Teacher Student

Un'ulteriore sperimentazione ha riguardato l'architettura di tipo Teacher-Student, un approccio avanzato basato sulla distillazione del modello in cui una rete neurale più complessa (Teacher) guida l'addestramento di una rete più leggera (Student). In questo caso, la rete Teacher coincide esattamente, sia in termini di struttura sia di funzione di perdita, con il modello descritto in precedenza per la classificazione dei segni: include una combinazione di Cross Entropy Loss e Contrastive o Triplet Loss, finalizzata all'apprendimento di embedding discriminativi.

La rete Student, invece, presenta un'architettura simile ma semplificata: il numero di layer nella parte convoluzionale è ridotto e la dimensione degli embedding è inferiore rispetto alla Teacher, con l'obiettivo di limitare la capacità espressiva del modello. L'addestramento della Student avviene tramite una Distillation Loss, che combina due componenti: una Cross Entropy Loss rispetto alle etichette reali (come in un training supervisionato tradizionale) e una Cross Entropy Loss rispetto ai logit prodotti dalla rete Teacher, in modo da guidare la rete più semplice a imitare il comportamento della più complessa sui dati considerati normali.

$$\mathcal{L}_{\text{distill}} = \alpha \cdot T^2 \cdot \text{KL} \left(\sigma \left(\frac{\mathbf{z}_s}{T} \right) \parallel \sigma \left(\frac{\mathbf{z}_t}{T} \right) \right) + (1 - \alpha) \cdot \mathcal{L}_{\text{CE}}(\mathbf{y}_s, \mathbf{y}_{\text{true}}) \quad (4.5)$$

L'ipotesi è che la rete Student, avendo una capacità inferiore, sia costretta a modellare fedelmente la distribuzione dei dati in-distribution per imitare il Teacher, ma fallisca nel generalizzare a input anomali, causando una divergenza misurabile nelle predizioni.

4.4. Classificazione Multisegno

Poiché l'obiettivo finale del progetto è la traduzione automatica, in tempo reale, di sequenze di segni della Lingua dei Segni Italiana (LIS), si rende necessario superare l'approccio tradizionale basato sulla classificazione di singoli gesti isolati, per affrontare il problema più complesso dell'interpretazione di sequenze continue, in cui i confini tra un segno e il successivo non sono noti a priori.

Questo scenario rispecchia in modo più fedele le condizioni d'uso reali, caratterizzate da una transizione fluida tra i segni, senza marcature esplicite di inizio e fine.

È opportuno precisare che, anche in questo contesto, quando si parla dell'input che viene passato ai modelli non si intende il video grezzo, bensì una sequenza di 42 caratteristiche numeriche (*features*) estratte mediante l'elaborazione con MediaPipe[40], come descritto nella sezione 4.1.

Per affrontare in maniera efficace questa sfida, sono stati analizzati e confrontati tre approcci distinti, ciascuno basato su una strategia differente per la classificazione di più segni all'interno di un singolo *stream video*.

4.4.1. Baseline: Classificazione con Sliding Window

Questo primo approccio può essere considerato come un'estensione diretta del paradigma della classificazione di segni isolati, adattato al contesto di una sequenza continua di segni. Si tratta di un metodo *naive*, che funge da *baseline* per il confronto con le soluzioni più avanzate analizzate nelle sezioni successive.

In particolare, il metodo si basa sulle migliori architetture identificate nella sezione 4.2.2, le quali sono state inizialmente addestrate su un *dataset* di segni isolati. Ciascun modello riceve in input sequenze di 150 frame, in cui eventuali frame mancanti vengono riempiti tramite *padding*, e viene ottimizzato utilizzando una funzione di perdita che combina *triplet loss* e *cross-entropy*.

La principale differenza rispetto al caso di classificazione di singoli segni risiede nella fase di inferenza. Nel contesto originale, il modello riceveva come input un segmento di frame contenente un singolo segno, mentre qui l'input consiste in un flusso continuo di *features*. Si rende quindi necessario sezionare il flusso in finestre temporali su cui applicare il modello di classificazione. A tal fine, è stata adottata una tecnica di *sliding window*, nella quale sia la dimensione della finestra w sia lo *stride* tra finestre consecutive sono stati ottimizzati mediante Optuna.

La lunghezza massima della finestra è stata vincolata a quella utilizzata durante la fase di training (5 secondi di video), soglia comunque superiore a quanto richiesto per la maggior parte dei segni. Qualora la finestra selezionata contenga meno frame rispetto al massimo consentito, essa viene completata con zeri per garantire la compatibilità con l'architettura di rete.

Per la valutazione del metodo, è stato utilizzato un insieme di video contenenti sequenze di segni già presenti nel *training set*. Infine, per migliorare la leggibilità dell'output, è stata applicata una fase di post-elaborazione finalizzata a comprimere eventuali sequenze di predizioni identiche consecutive: se un segno viene ripetuto in maniera contigua, esso viene conteggiato come un'unica occorrenza.

4.4.2. Secondo Approccio: Classificazione con Finestre di Dimensione Ridotta e *Major Voting*

Il secondo approccio nasce dall'esigenza di ridurre la discrepanza tra le modalità di utilizzo del modello in fase di addestramento e in fase di inferenza, criticità presente nel metodo descritto nella sezione precedente. L'obiettivo è stato quindi quello di far sì che la rete elaborasse, in entrambe le fasi, dati caratterizzati da proprietà statistiche il più possibile simili.

A tale scopo, si è scelto di utilizzare finestre temporali di dimensioni inferiori rispetto al primo approccio. La dimensione della finestra w è stata selezionata in modo da risultare sufficientemente ampia da catturare le informazioni fondamentali del segno, ma al contempo abbastanza ridotta da minimizzare la presenza di più segni all'interno della stessa finestra.

Durante la fase di inferenza, anche in questo caso il flusso di *features* viene suddiviso in segmenti tramite una strategia di *sliding window*, con finestra w e *stride* s . Tuttavia, viene introdotto un meccanismo più sofisticato: ogni finestra w può contenere più frame rispetto a quelli previsti come input dal modello (i). Per tale ragione, da ciascuna finestra vengono estratte $w - i + 1$ sotto-sequenze di lunghezza i , le cui predizioni vengono poi combinate tramite una procedura di *major voting*. In questo modo si migliora la capacità della finestra di individuare correttamente il segno prevalente nello *stream*.

Anche il *dataset* di addestramento è stato adattato di conseguenza: le sequenze di frame, appartenenti al *dataset* composto da 72 classi descritto in 4.1 e contenenti ciascuna un singolo segno, sono state frammentate in finestre della stessa dimensione w adottata in fase di inferenza. I modelli sono stati addestrati utilizzando una funzione di perdita combinata,

basata su *triplet loss* e *categorical cross-entropy*, con l'obiettivo di massimizzare sia la separabilità tra le classi che la precisione nella classificazione.

Le architetture impiegate in questo approccio comprendono modelli convoluzionali e reti LSTM, strettamente correlate a quelle analizzate nella sezione 4.2.2, ma con un *input layer* di dimensioni ridotte per adattarsi alle nuove finestre temporali.

Infine, come nel primo approccio, è stato applicato un processo di post-elaborazione per aggregare le predizioni consecutive appartenenti alla stessa classe, al fine di migliorare la leggibilità dell'output finale.

4.4.3. Utilizzo di modelli LSTM end-to-end

In alternativa ai metodi basati su segmentazione esplicita, è stato esplorato un approccio di tipo *end-to-end* basato su modelli *Long Short-Term Memory* (LSTM), capaci di ricevere in input l'intera sequenza video e di restituire direttamente la serie di segni riconosciuti. Tali modelli sono stati preferiti ai *Transformers* in quanto, in scenari caratterizzati da dati limitati, offrono una maggiore accuratezza e consentono di modellare in maniera naturale le dipendenze temporali a lungo termine tra i segni, evitando la necessità di una segmentazione preliminare.

Poiché non era disponibile un dataset di sequenze continue di segni, si è reso necessario costruire un dataset sintetico a partire dal dataset di segni isolati descritto in 4.1. A tal fine, sono state concatenate in maniera casuale e bilanciata sequenze composte da un numero variabile di segni, compreso tra 4 e 6, selezionati randomicamente. Tale procedura ha permesso di simulare la continuità temporale del linguaggio dei segni, preservando al contempo la variabilità del dataset originale. Inoltre, per ridurre l'impatto delle discontinuità ai confini tra segni adiacenti, sono stati introdotti frame sintetici di transizione, con l'obiettivo di rendere le sequenze più realistiche e favorire l'apprendimento del modello. Contestualmente, sono state generate etichette sintetiche corrispondenti all'unione dei segni presenti nella sequenza considerata, al fine di consentire l'addestramento supervisionato dell'intero modello *end-to-end*.

Le architetture LSTM sperimentate differiscono tra loro per diversi aspetti progettuali: alcune includono meccanismi di *attention* per migliorare la capacità di modellare contesti temporali estesi, altre variano in termini di complessità strutturale (numero di layer, dimensione degli *hidden states*) e funzione di perdita adottata. Questa fase di analisi ha permesso di valutare l'impatto delle diverse configurazioni sull'accuratezza e la capacità di generalizzazione del modello.

Nonostante il potenziale di questo approccio, l'addestramento di modelli *end-to-end* richiede una notevole quantità di dati annotati con precisione e un attento controllo dei fenomeni di *overfitting*, specialmente in presenza di sequenze sintetiche e nel caso di generalizzazione verso sequenze non viste.

5 | Risultati

In questo capitolo vengono presentati e analizzati in dettaglio i risultati sperimentali ottenuti dalle metodologie descritte nel capitolo precedente. L'obiettivo è duplice: in primo luogo, quantificare le performance di ciascun approccio sviluppato e, in secondo luogo, effettuare un'analisi comparativa per identificare le architetture e le strategie più efficaci per la traduzione automatica della Lingua dei Segni Italiana.

La presentazione dei risultati seguirà una progressione logica, rispecchiando la struttura del capitolo dei metodi per garantire la massima chiarezza. Si partirà dalla valutazione del task fondamentale della classificazione di un singolo segno isolato, confrontando modelli di machine learning tradizionali con architetture di deep learning avanzate quali 1D-CNN, LSTM e Transformer. Le performance saranno misurate attraverso metriche standard come Accuratezza, Precisione, Recall e F1-Score, e verranno analizzate tramite matrici di confusione per identificare specifici pattern di errore.

Successivamente, verranno esposti i risultati del modulo di Anomaly Detection, cruciale per la robustezza del sistema in scenari reali. L'efficacia dei diversi metodi nel riconoscere segni non appartenenti al vocabolario noto verrà valutata principalmente tramite la metrica Area Under the Curve (AUC-ROC).

Infine, il capitolo affronterà la sfida più complessa della classificazione di sequenze multisegno. Verranno confrontate le tre strategie proposte — finestre fisse con raffinamento probabilistico, finestre variabili con filtro di anomalie e modelli end-to-end — utilizzando metriche specifiche per l'analisi di sequenze, come il Word Error Rate (WER). L'analisi quantitativa sarà arricchita da esempi qualitativi di traduzione per fornire una valutazione completa della capacità di generalizzazione e dell'efficacia pratica del sistema finale.

5.1. Risultati della Classificazione di Segni Singoli

Per valutare le prestazioni relative a questo *task*, è stato adottato come principale indicatore l'accuratezza. I *dataset* utilizzati, descritti nella Sezione 4.1, sono due: il primo comprende 46 classi e il secondo 72 classi. Entrambi sono stati suddivisi nella proporzione

80%-10%-10% rispettivamente per *training*, *validation* e *test set*. I risultati riportati di seguito si riferiscono esclusivamente al *test set*.

5.1.1. Performance dei Modelli Baseline

In una fase iniziale, il problema è stato affrontato mediante l'impiego di metodi classici di *Machine Learning*, nello specifico *Logistic Regression*, *Support Vector Machine* (SVM), *Decision Trees* e *Random Forest*. Nonostante la complessità intrinseca del compito di classificazione, tali modelli hanno dimostrato prestazioni complessivamente solide, con risultati proporzionali alla capacità di generalizzazione e alla complessità architettonica di ciascun approccio.

È stata osservata una differenza prestazionale significativa tra il dataset a 46 classi e quello a 72 classi, imputabile alla maggiore difficoltà introdotta dall'aumento del numero di categorie e, conseguentemente, alla maggiore variabilità intra-classe e alla ridotta separabilità tra classi.

Tra i modelli considerati, le *Random Forest* hanno ottenuto le migliori prestazioni, raggiungendo un'accuratezza pari al 92.4% sul dataset a 46 classi e all'82.7% sul dataset a 72 classi.

Per completare il confronto e valutare le prestazioni di una rete neurale a bassa complessità architettonica, è stato inoltre sperimentato un *Multilayer Perceptron* (MLP) costituito da uno strato di input, due strati nascosti composti rispettivamente da 512 e 128 neuroni, e uno strato di output. Il modello ha evidenziato performance leggermente superiori rispetto agli algoritmi di *Machine Learning* più semplici, mostrando tuttavia risultati inferiori a quelli ottenuti dalla *Random Forest*.

Tabella 5.1: Prestazioni dei modelli sui due dataset (46 e 72 classi) in termini di Accuracy, Weighted Precision e Weighted F1-score.

Modelli ML	46 classi			72 classi		
	Acc.	Prec.	F1	Acc.	Prec.	F1
Logistic Regression	84.5	85.4	84.6	66.0	66.8	65.9
SVM	87.7	88.5	87.8	71.4	73.2	71.6
Decision Tree	75.1	75.3	74.8	50.5	49.6	47.9
Random Forest	92.4	92.9	92.5	82.7	83.4	82.4
Multilayer Perceptron	87.1	87.8	87.0	73.2	73.5	72.9

5.1.2. Confronto tra Architetture di Deep Learning Avanzate

Dopo la definizione della baseline, sono state esplorate soluzioni appartenenti al campo del *deep learning*, con l'obiettivo di individuare modelli in grado di catturare in maniera più efficace la complessità temporale e spaziale dei dati di input. In particolare, sono stati analizzati tre approcci principali: reti convoluzionali monodimensionali (1D-CNN), reti ricorrenti basate su LSTM e modelli Transformer.

Per tutte le architetture è stata condotta una fase di *hyperparameter tuning* mediante Optuna, strumento di ottimizzazione automatica. Sono stati esplorati parametri quali il tasso di dropout, il *learning rate*, il numero di unità nei livelli nascosti e la profondità della rete. L'ottimizzazione è stata guidata dalla *categorical cross entropy* come funzione di perdita, con l'impiego di *early stopping* per prevenire fenomeni di overfitting.

I risultati evidenziano come tutte e tre le architetture superino le prestazioni della baseline, confermando che i modelli di *deep learning*, grazie alla loro maggiore capacità rappresentativa, riescono a catturare più efficacemente la complessità dei dati rispetto ai metodi tradizionali di *machine learning*. Tra i modelli analizzati, le reti LSTM si sono dimostrate le più efficaci, raggiungendo un'accuratezza pari a **95.4%** sul dataset con 46 segni e **91.1%** sul dataset con 72 segni. Come riportato nella Tabella 5.2, i tre modelli presentano prestazioni molto simili sul dataset a 46 classi, mentre le differenze diventano più marcate nel caso del dataset a 72 classi. Questo risultato conferma l'importanza di un'adeguata modellazione della componente temporale per il riconoscimento automatico della LIS. I grafici riportati nelle figure seguenti illustrano l'andamento della ricerca degli iperparametri e le curve di addestramento dei modelli.

Tabella 5.2: Prestazioni dei modelli di Deep Learning addestrati con la *categorical cross entropy* sui due dataset (46 e 72 classi), riportate in termini di Accuracy, Weighted Precision e Weighted F1-score.

Reti Neurali	46 classi			72 classi		
	Acc.	Prec.	F1	Acc.	Prec.	F1
1D-CNN	94.4	94.8	94.6	86.2	86.3	85.4
LSTM	95.4	95.7	95.6	91.1	91.2	90.9
TRANSFORMER	95.2	95.3	95.2	90.9	90.8	90.7

Analizzando le matrici di confusione emerge come tutte le classi presenti nei due dataset, sia quello a 46 che quello a 72 segni, siano state complessivamente apprese in maniera corretta. La predominanza dei valori lungo la diagonale conferma infatti la capacità dei

Training BiLSTM

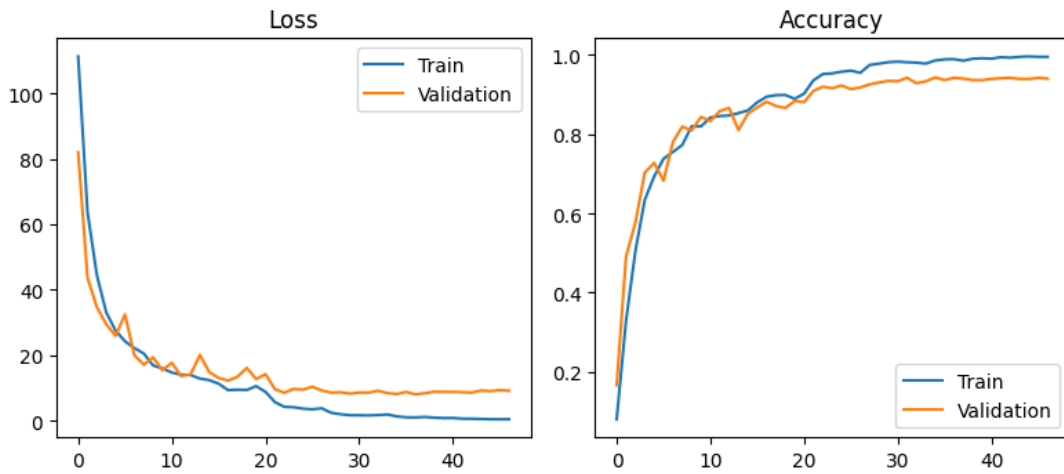


Figura 5.2: Training Loss (46 classi)

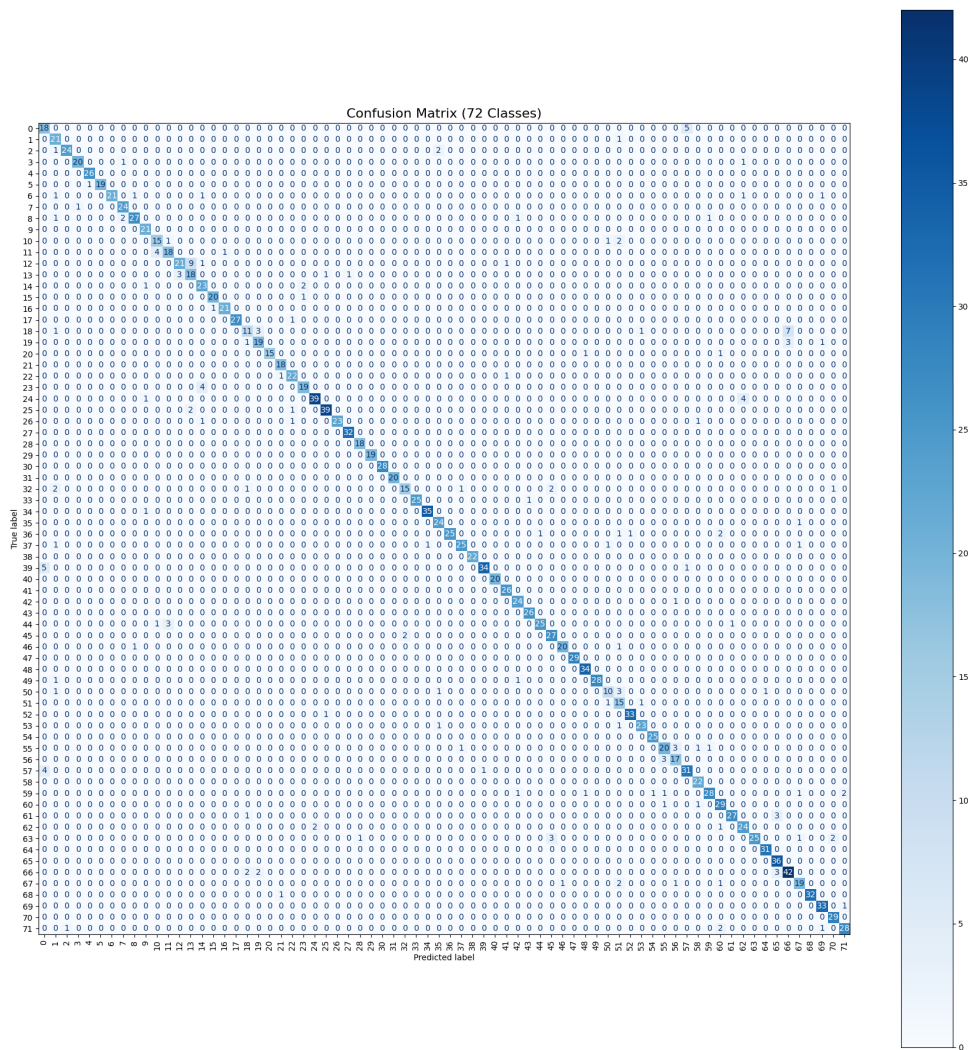


Figura 5.3: Confusion Matrix (72 classi)

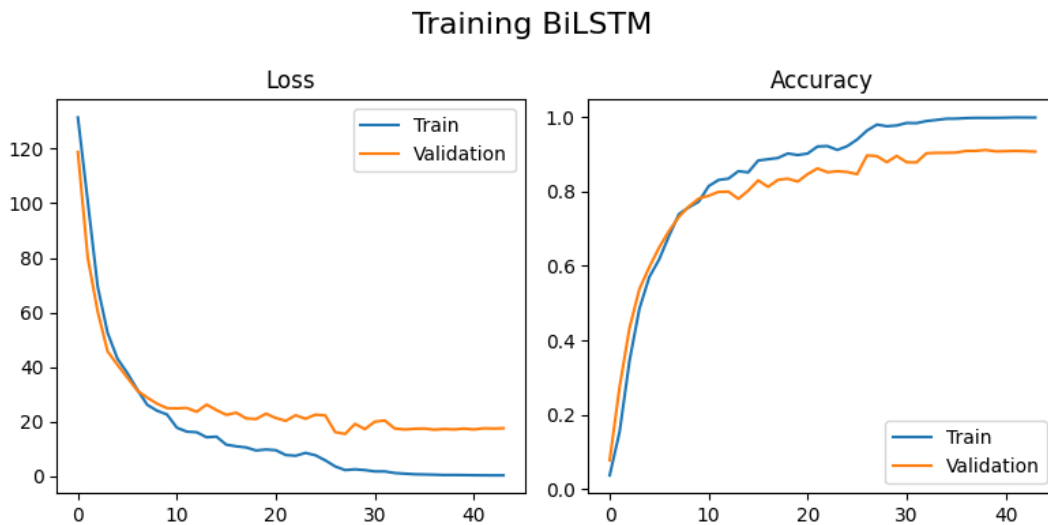


Figura 5.4: Training Loss (72 classi)

5.1.3. Impatto delle Funzioni di Loss

Un elemento cruciale nell'addestramento delle reti neurali è la scelta della funzione di *loss*, che rappresenta la vera e propria bussola del processo di ottimizzazione: ogni parametro del modello viene infatti aggiornato in funzione del gradiente calcolato rispetto a questa funzione. Nei problemi di classificazione, lo standard è rappresentato dalla *categorical cross entropy*, che costituisce la scelta di riferimento anche nel caso del presente lavoro.

Tuttavia, la letteratura mostra come la combinazione della *categorical cross entropy* con altre funzioni di perdita possa migliorare la qualità delle rappresentazioni apprese. In particolare, sono state prese in considerazione la *Siamese Loss* e la *Triplet Loss*, entrambe progettate per favorire una migliore separazione nello spazio degli embedding.

Come evidenziato nei grafici (Fig.5.5a), l'impiego di queste funzioni consente di forzare esplicitamente una maggiore distanza tra le rappresentazioni appartenenti a classi differenti, riducendo così le ambiguità tra categorie spesso confuse. L'analisi delle matrici di confusione, ottenute dai modelli descritti nella sezione precedente, conferma un miglioramento nella separazione tra le classi più problematiche. È tuttavia necessario combinare tali funzioni con la *categorical cross entropy*, al fine di garantire che anche gli strati finali del modello possano essere correttamente addestrati per la predizione delle etichette.

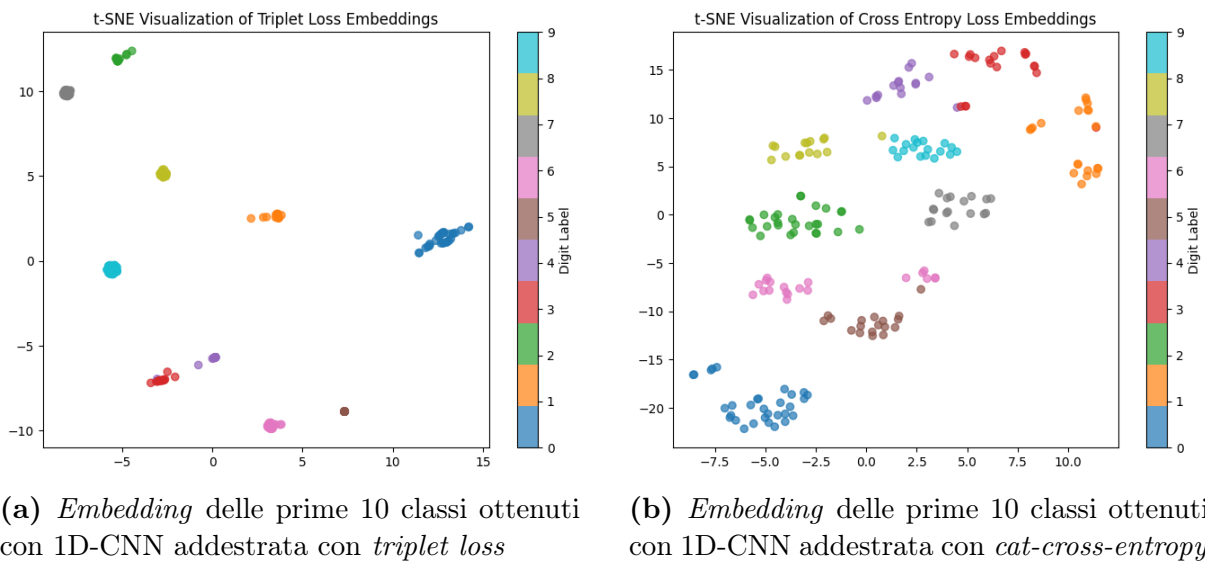


Figura 5.5: Confronto tra le rappresentazioni nello spazio degli *embedding* generate da una 1D-CNN addestrata con due diverse funzioni di perdita.

Anche in questa fase, è stata condotta una ricerca degli iperparametri focalizzata principalmente sulla scelta delle funzioni di loss e degli ottimizzatori, piuttosto che sulla struttura dei modelli. Le architetture di riferimento sono infatti quelle risultate ottimali dalla fase di ottimizzazione eseguita con Optuna, descritta in precedenza. Come riportato nella Tabella 5.3, la combinazione della *categorical cross entropy* con funzioni come la *Siamese Loss* e la *Triplet Loss* ha permesso di ottenere un incremento di alcuni punti percentuali in termini di accuratezza. Tale miglioramento comporta, tuttavia, un incremento significativo dei tempi di addestramento, dovuto sia al maggior numero di epoche necessarie per il completamento del training, sia al fatto che le chiamate al modello raddoppiano con la *Siamese Loss* e triplicano con la *Triplet Loss*. In questi casi, infatti, a ciascuna immagine vengono associate una o due controparti per il confronto, con un conseguente aumento del costo computazionale.

Dai risultati riportati in Tabella 5.3 emerge come tutte e tre le architetture considerate beneficiano dell'introduzione di funzioni di perdita aggiuntive rispetto alla sola *categorical cross entropy*. In particolare, l'impiego della *Triplet Loss* porta mediamente a prestazioni leggermente superiori rispetto alla *Siamese Loss*, sebbene le differenze non siano sempre significative e risultino in parte dipendenti dal dataset e dall'architettura adottata.

Nel complesso, si può osservare come tutte e tre le architetture abbiano fatto registrare prestazioni molto simili, con differenze contenute sia in termini di accuratezza che di metriche derivate. Tra esse, la *LSTM* si è confermata la più efficace, seguita a breve distanza dalla *1D-CNN* e dal *Transformer*. Questo risultato evidenzia come, pur a fronte

Tabella 5.3: Prestazioni dei modelli di Deep Learning con diverse funzioni di loss sui due dataset (46 e 72 classi), in termini di Accuracy, Weighted Precision e Weighted F1-score.

Modello + Loss	46 classi			72 classi		
	Acc.	Prec.	F1	Acc.	Prec.	F1
1D-CNN + CE+Siamese	95.47	95.7	95.7	89.1	88.8	88.5
1D-CNN + CE+Triplet	96.0	96.28	96.2	90.37	90.1	89.8
LSTM + CE+Siamese	95.5	95.8	95.7	91.6	91.6	91.3
LSTM + CE+Triplet	96.95	97.1	97.1	92.9	92.9	92.6
Transformer + CE+Siamese	96.40	96.57	96.59	92.7	92.5	92.4
Transformer + CE+Triplet	96.4	96.6	96.6	92.0	92.1	91.6

di approcci architetturali differenti, l'influenza della funzione di perdita risulti in alcuni casi più determinante della scelta del modello stesso.

5.2. Risultati Out Of Distribution

La natura peculiare delle lingue dei segni, unita al numero limitato di segni considerati durante la fase di addestramento, rende necessario valutare strategie che possano migliorare la robustezza delle soluzioni proposte.

In particolare, diventa cruciale introdurre strategie di out-of-distribution (OOD) detection, al fine di distinguere tra segni appartenenti al vocabolario di addestramento e segni non precedentemente osservati.

Questa sezione è pertanto dedicata all'analisi dell'efficacia dei metodi esposti nella sezione 4.3 per rilevare e gestire segni della LIS non inclusi nel set di addestramento, con l'obiettivo di migliorare la robustezza e l'affidabilità complessiva della soluzione proposta.

Le analisi sono state condotte utilizzando il dataset composto da 46 segni. Per simulare la presenza di elementi mai osservati, il dataset è stato suddiviso in due parti: un insieme di segni noti, costituito da 40 classi, e un insieme di segni non noti, formato da 6 classi selezionate casualmente e trattate come *out-of-distribution*.

5.2.1. CNN extraction + anomaly detection algorithms

In questa sezione viene presentata una prima soluzione al problema dell'identificazione di segni LIS *out-of-distribution* (OOD) rispetto al dataset di addestramento. L'obiettivo

non è quello di proporre un metodo allo stato dell'arte, bensì di definire una baseline di riferimento che consenta di evidenziare i limiti degli approcci tradizionali e motivare l'impiego di strategie più avanzate nelle sezioni successive.

La soluzione si basa su un approccio in due fasi. Nella prima fase, gli *embedding* vengono estratti mediante una rete neurale convoluzionale addestrata come descritto nella Sezione 5.1.2, utilizzando una funzione di perdita combinata (*categorical cross-entropy* e *triplet loss*) per favorire una migliore separazione nello spazio delle rappresentazioni. Successivamente, tali *embedding* vengono analizzati tramite algoritmi classici di *anomaly detection*, quali DBSCAN, K-means e Isolation Forest, con l'obiettivo di discriminare tra esempi appartenenti al dominio di addestramento e potenziali istanze OOD.

Gli *embedding* dei segni *in-distribution* tendono a organizzarsi in cluster relativamente compatti, ma non sufficientemente distinti né distanziati rispetto alle rappresentazioni dei segni *out-of-distribution*. Ciò genera regioni di sovrapposizione, all'interno delle quali diventa difficile distinguere in maniera affidabile le istanze OOD.

Per valutare le prestazioni degli algoritmi considerati, è stata condotta una fase di ricerca degli iperparametri mediante *Optuna*, ottimizzando i parametri caratteristici di ciascun metodo di *anomaly detection*. I risultati, riportati nella Tabella 5.4, mostrano come nessuno degli approcci raggiunga performance particolarmente elevate; tuttavia, emergono differenze significative tra i metodi.

In particolare, l'algoritmo **Isolation Forest** si dimostra il più efficace, con un'accuratezza del 67% e un F1-score OOD pari al 65,2%. Tali valori superano di circa 6 punti percentuali le prestazioni ottenute da **K-means**, che si colloca in posizione intermedia con il 60,5% di accuratezza e un F1-score del 57,1%. Il metodo **DBSCAN**, invece, evidenzia i risultati più limitati, con un'accuratezza del 57,0% e un F1-score pari al 49,2%.

Questa tendenza è coerente con la natura degli algoritmi: **Isolation Forest**, basato su partizionamenti ricorsivi dello spazio, è più flessibile nell'individuare anomalie anche in distribuzioni complesse, mentre **K-means**, vincolato a cluster di forma sferica e sensibile al valore di k , offre prestazioni più contenute. **DBSCAN**, infine, risente della difficoltà di individuare densità eterogenee negli embedding, aspetto che ne riduce l'efficacia.

Nonostante ciò, le performance complessive rimangono moderate, a indicare che gli embedding estratti dalla CNN non risultano ancora sufficientemente discriminativi per supportare appieno l'*anomaly detection*. In tale contesto, gli algoritmi classici di *anomaly detection*, applicati a queste rappresentazioni, non si dimostrano in grado di garantire una separazione efficace tra classi *in-distribution* e istanze OOD. Pur costituendo un utile

Tabella 5.4: Prestazioni in % dei metodi di anomaly detection applicati agli *embedding* estratti tramite CNN, in termini di AUROC, ACC e F1-score OOD.

Metodo	ACC	PREC	F1 OOD
CNN + Isolation Forest	67.0	68.8	65,2
CNN + DBSCAN	57.0	57.0	49.2
CNN + K-means	60.5	58.8	57.1

punto di partenza, essi presentano limiti strutturali che rendono necessaria l'adozione di soluzioni più sofisticate, capaci di modellare in maniera esplicita la complessità dei dati.

5.2.2. Valutazione degli Approcci basati su Logit

Un secondo approccio si fonda sull'assunzione che, quando la rete si trova di fronte a un segno mai osservato in fase di addestramento, le sue predizioni tendano a risultare incerte. Per quantificare tale incertezza sono stati adottati tre criteri differenti: il valore massimo della distribuzione predittiva (*Max Probability*), l'entropia della distribuzione predittiva e la funzione di energia (*Energy*) [4.3.2].

In ciascuno di questi casi sono stati introdotti due parametri fondamentali: un fattore di temperatura, utilizzato per scalare le probabilità derivate dai logit, e una soglia decisionale, necessaria per distinguere i campioni *in-distribution* da quelli *out-of-distribution* (OOD). La ricerca dei valori ottimali per tali parametri è stata condotta mediante *hyperparameter search* con Optuna, utilizzando la *cross-validation* sul dataset, composto da 40 classi di segni in-distribution e 6 classi out-of-distribution. Le configurazioni migliori sono state successivamente valutate sul *test set* per ottenere una stima affidabile delle prestazioni.

Le sperimentazioni sono state condotte impiegando le migliori architetture individuate nella Sezione 5.1.2, addestrate esclusivamente sul sottoinsieme dei 40 segni *in-distribution*.

L'analisi dei risultati riportati nelle Tabelle 5.5, 5.6 e 5.7 mette in evidenza alcune tendenze. Innanzitutto, si osserva come l'architettura **LSTM** ottenga sistematicamente le prestazioni migliori in termini di AUROC (fino a 0.8705 con il criterio dell'entropia) e di accuratezza OOD (76.5%), confermandosi particolarmente adatta a catturare le dipendenze temporali presenti nei dati. L'approccio **Transformers** si colloca in posizione intermedia, con valori di AUROC compresi tra 0.7930 e 0.8030 e accuratezze tra il 67% e il 70%, mentre la **1D CNN** mostra le performance più contenute, soprattutto per quanto riguarda la capacità discriminativa (AUROC tra 0.6840 e 0.7520).

Dal punto di vista dei criteri adottati, i risultati evidenziano come l'**entropia** rappresenti il metodo più robusto, portando a un miglioramento consistente rispetto al semplice *Max Probability*, mentre la funzione di **energia** offre prestazioni comparabili ma leggermente inferiori. Questo conferma che strategie capaci di modellare esplicitamente l'incertezza predittiva risultano più efficaci per distinguere campioni in-distribution da istanze OOD.

In generale, pur evidenziando differenze tra architetture e criteri, le performance ottenute restano moderate: gli algoritmi logit-based riescono a catturare parzialmente l'incertezza associata a campioni OOD, ma non garantiscono ancora una separazione netta e affidabile.

Tabella 5.5: Risultati OOD basati su *Max Probability* per le diverse architetture.

Architettura	Best Threshold	Temperature	AUROC	Accuracy OOD
1D CNN	4.043	2.565	0.6840	63.0%
LSTM	15.844	2.968	0.8315	76.5%
Transformers	13.874	1.405	0.8030	67.0%

Tabella 5.6: Risultati OOD basati su *Entropy* per le diverse architetture.

Architettura	Best Threshold	Temperature	AUROC	Accuracy OOD
1D CNN	1.887	2.135	0.7520	66.5%
LSTM	2.863	4.620	0.8705	76.5%
Transformers	0.259	1.462	0.7930	70.0%

Tabella 5.7: Risultati OOD basati su *Energy* per le diverse architetture.

Architettura	Best Threshold	Temperature	AUROC	Accuracy OOD
1D CNN	-4.194	1.198	0.6905	64.0%
LSTM	-16.840	3.154	0.8435	76.0%
Transformers	-14.031	1.222	0.8030	67.0%

5.2.3. Approcci Teacher-Student

In questo approccio è stato adottato un paradigma *teacher-student*, che prevede l'impiego di due architetture: una *teacher*, più complessa e accurata, e una *student*, più leggera e allenata a partire dalle predizioni del modello teacher. L'ipotesi alla base è che, per campioni appartenenti alla distribuzione di addestramento (*in-distribution*), le due reti producano predizioni simili, mentre in presenza di campioni *out-of-distribution* emergano discrepanze più marcate.

Come modelli teacher sono state impiegate le migliori architetture individuate nella sezione 5.1.2, mentre i corrispondenti modelli student sono stati realizzati come versioni semplificate delle rispettive reti teacher. La rilevazione OOD è stata effettuata applicando una soglia sul grado di divergenza tra le predizioni delle due reti. Tale valore soglia è stato ottimizzato massimizzando la *Youden’s J statistic*, definita come differenza tra il tasso di veri positivi e quello di falsi positivi, utilizzando come misura di divergenza la *Kullback–Leibler Divergence*.

Dai risultati ottenuti, riportati nella Tabella 5.8, emerge che l’approccio risulta efficace nel discriminare tra segni appartenenti e non appartenenti alla distribuzione di addestramento, anche se le performance ottenute non superano l’80% di accuratezza.

Tabella 5.8: Confronto tra modelli Teacher e Student con differenti strategie di addestramento.

Architettura	Teacher Acc. Trpl	Student Acc. Trpl	Student Acc. Distill
1D CNN	95.7%	86.5%	89.5%
LSTM	95.0%	91.0%	92.5%
Transformers	94.9%	91.50%	94.0%

Tabella 5.9: Risultati di rilevazione OOD per le diverse architetture.

Architettura	AUROC	TPR	Accuracy OOD
1D CNN	0.7760 ± 0.0583	0.67	78.5%
LSTM	0.6200 ± 0.0677	0.73	66.6%
Transformers	0.7950 ± 0.0232	0.85	78.5%

L’analisi dei risultati riportati nelle Tabelle 5.8 e 5.9 consente di trarre alcune importanti considerazioni. In primo luogo, si osserva che l’addestramento tramite *knowledge distillation* porta a un miglioramento consistente delle prestazioni degli studenti rispetto al semplice addestramento tradizionale: ad esempio, la 1D CNN passa dall’86.5% all’89.5% di accuratezza, mentre i Transformers raggiungono il 94.0%, avvicinandosi al livello del rispettivo teacher. Questo conferma l’efficacia della distillazione come strategia di trasferimento delle conoscenze da modelli complessi a reti più leggere.

Per quanto riguarda la rilevazione OOD, i risultati evidenziano una marcata variabilità tra le architetture. L’approccio basato sui **Transformers** ottiene le performance migliori, con un AUROC pari a 0.7950 e un tasso di veri positivi dell’85%, mentre la **1D CNN** raggiunge valori comparabili (AUROC = 0.7760, TPR = 0.67). Al contrario, l’**LSTM** mostra risultati più modesti (AUROC = 0.6200), segnalando una minore capacità di discriminare efficacemente le istanze OOD.

Nel complesso, l'approccio *teacher-student* si dimostra promettente sia per la riduzione della complessità dei modelli, grazie a reti *student* più leggere ma accurate, sia per la capacità di supportare la rilevazione OOD. Tuttavia, le performance ottenute non superano il 79% di accuratezza OOD, suggerendo che, pur valido, questo metodo non è sufficiente da solo a garantire una separazione netta e affidabile tra campioni in-distribution e out-of-distribution.

5.3. Risultati della Classificazione Multisegno

La terza fase sperimentale, dedicata alla classificazione multisegno, ha costituito l'aspetto più complesso dell'intero progetto, poiché mirava alla traduzione di sequenze continue di segni della Lingua dei Segni Italiana (LIS) nelle corrispondenti unità testuali. L'obiettivo principale è stato valutare l'efficacia di diversi approcci nell'affrontare le difficoltà intrinseche di questo compito, tra cui l'elevata variabilità dei movimenti, la gestione delle transizioni tra segni consecutivi e la presenza di rumore nei dati acquisiti.

A tal fine è stato predisposto un test set specifico, contenente video caratterizzati da sequenze continue di segni. Alcune di queste sequenze rappresentano l'intero alfabeto, mentre altre includono serie di segni appartenenti al set di addestramento, così da garantire un'adeguata varietà e complessità nella fase di valutazione.

La performance dei modelli è stata misurata attraverso metriche standard, tra cui *precision*, che indica la proporzione di segni predetti correttamente rispetto al totale delle predizioni, e *recall*, che misura la capacità del modello di individuare correttamente i segni effettivamente presenti. A queste si affianca l'*F1-score*, che sintetizza in un'unica misura *precision* e *recall*, insieme a metriche specifiche come *WREC* utile per una valutazione più approfondita della qualità complessiva della traduzione automatica.

Nelle sezioni seguenti vengono presentati e discussi in dettaglio i risultati ottenuti dai tre approcci sperimentali 4.4 adottati per la classificazione multisegno, con particolare attenzione ai punti di forza e alle criticità emerse nell'analisi comparativa.

5.3.1. Risultati del Metodo Baseline

Il primo approccio sperimentale rappresenta una baseline, concepita come estensione diretta della classificazione di segni isolati al caso più complesso delle sequenze continue. L'obiettivo principale di questo metodo era verificare se fosse possibile trasferire, almeno in parte, le conoscenze acquisite nella classificazione di singoli segni alla nuova sfida della classificazione multisegno, caratterizzata da una diversa natura dell'input e da una

complessità temporale significativamente superiore.

I modelli impiegati comprendono tre architetture rappresentative: LSTM, CNN e Transformer. Tutti sono stati addestrati utilizzando una funzione di perdita che combina *triplet loss* e *cross-entropy*, al fine di coniugare capacità discriminativa e robustezza nella fase di classificazione. Le architetture selezionate corrispondono alle migliori configurazioni individuate nella sezione 5.1.2.

L'addestramento di tali architetture è stato condotto utilizzando come dataset di *training* quello descritto in 4.1, in cui ciascun campione corrisponde a un singolo segno ed è rappresentato da una sequenza di 150 frame. Le sequenze con un numero di frame inferiore sono state completate tramite *padding*, così da garantire uniformità nella lunghezza degli input forniti al modello.

La valutazione dei metodi è stata effettuata attraverso le metriche di precision, recall, F1-score e WER, riportate in dettaglio nelle tabelle 5.10. I dati impiegati nella fase di test provengono dal set di segni continui descritto in 5.3, le cui sequenze, per poter essere processate dal modello, sono state necessariamente suddivise mediante l'applicazione della tecnica di *sliding window*, con un'ampiezza di 30 frame, corrispondente a circa un secondo di video.

L'analisi dei risultati evidenzia come l'approccio basato su finestre temporali fisse mostri prestazioni significativamente limitate. In particolare, il modello non solo fatica a identificare correttamente i segni presenti nella sequenza, ma risulta anche incapace di individuare con precisione i punti di transizione tra un segno e l'altro. Questa debolezza si traduce in un'alta frequenza di errori, sia a livello di classificazione dei singoli segni, sia nella segmentazione temporale della sequenza.

Tali risultati confermano la natura di baseline di questo metodo, che si configura come un primo tentativo, utile principalmente per evidenziare le sfide poste dalla classificazione multisegno e per motivare la necessità di soluzioni più avanzate, analizzate nelle sezioni successive.

Tabella 5.10: Risultati del metodo baseline basato su finestre fisse per la classificazione multisegno.

Modello	Precision	Recall	F1-score	WER
LSTM	25,5	16,6	20,2	2,21
CNN	20,3	14,5	16,9	2,38
Transformer	24,9	11,0	15,3	2,17

5.3.2. Secondo Approccio: Classificazione con Finestre di Dimensione Ridotta e Major Voting

Questo secondo approccio nasce dalla necessità di garantire una maggiore coerenza tra i dati utilizzati in fase di training e quelli impiegati durante l’inferenza. Per ottenere ciò, il dataset di training e validation è stato derivato dal dataset di partenza descritto in 4.1, frammentando le sequenze tramite una *sliding window* e evitando le porzioni di zeri introdotte artificialmente per uniformare le dimensioni. La lunghezza della finestra (w) è stata ottimizzata tramite ricerca sperimentale ed è risultata pari a 15 frame, corrispondenti a circa mezzo secondo.

I modelli impiegati sono analoghi a quelli analizzati in 5.1.2, ma con un *input layer* riadattato per gestire sequenze di 15 frame. Durante la fase di inferenza, l’input continuo è stato segmentato mediante una *sliding window* di ampiezza maggiore rispetto a quella prevista dalla rete ($w = 20$, $\text{stride} = 10$), generando sotto-sequenze di lunghezza 15. Le predizioni associate a ciascuna sotto-sequenza vengono quindi combinate mediante un meccanismo di *major voting*, migliorando l’identificazione del segno prevalente all’interno della finestra.

Il training dei modelli è stato effettuato utilizzando una funzione di perdita combinata, basata su *triplet loss* e *categorical cross-entropy*, con l’obiettivo di massimizzare sia la separabilità tra le classi sia la precisione nella classificazione.

La valutazione delle prestazioni è stata condotta tramite le metriche di *precision*, *recall*, *F1-score* e *Word Error Rate* (WER), riportate in dettaglio nelle tabelle 5.11 5.12. I dati utilizzati nella fase di test provengono dal set di segni continui descritto in 5.3, sui quali è stata applicata la stessa strategia di segmentazione con *sliding window* e *major voting*.

I risultati ottenuti 5.12 mostrano un netto miglioramento rispetto alla baseline, dimostrando che la riduzione della discrepanza tra dati di training e dati estratti dallo stream di frame favorisce una maggiore accuratezza nella classificazione dei segni, in particolare per sequenze brevi come le lettere dell’alfabeto.

Tabella 5.11: Risultati dei modelli sul dataset di testing dei segni isolati di lunghezza 15 frame

Modello	Acc.	Prec.	F1-score
CNN	98,36	98,34	98,32
LSTM	99,93	99,93	99,93

Tabella 5.12: Risultati del metodo basato su finestre ridotte e Major Voting.

Modello	Prec.	Recall	F1-score	WER
CNN	66,1	58,2	61,9	1,08
LSTM	78,2	74,1	76,0	0,71

5.3.3. Performance dell'Approccio End-to-End

In quest'ultimo approccio sono state analizzate le prestazioni del modello BI-LSTM, arricchito con meccanismi di attenzione e addestrato sul dataset sintetico 4.4.3, con l'obiettivo di valutarne la capacità di catturare in modo efficace le dipendenze spazio-temporali presenti nelle sequenze di segni.

I risultati ottenuti sul dataset di test sintetico, riportati in Tabella 5.13, evidenziano come i modelli LSTM siano in grado di catturare efficacemente le dipendenze spazio-temporali tra i segni e di riconoscere correttamente le sequenze generate. In particolare, le architetture dotate di meccanismi di attenzione hanno mostrato prestazioni superiori, suggerendo che la capacità di pesare in modo dinamico i diversi frame della sequenza risulta cruciale per la corretta classificazione.

Quando il modello è stato valutato sugli stream video reali, i cui risultati sono mostrati in Tabella 5.13, si è osservato un leggero calo delle performance rispetto al dataset sintetico. Questo scostamento può essere attribuito al bias introdotto dalla procedura di creazione del dataset artificiale, che, pur riducendo il rumore e la variabilità presente nei dati reali, non ne riproduce fedelmente la complessità.

Tabella 5.13: Risultati del migliore modello end-to-end sul dataset di testing

Dataset	Prec.	Recall	F1-score
Sintetico	97,6	96,8	97,1
Reale	70,3	69,4	69,8

6 | Discussione dei Risultati

I risultati presentati nei capitoli precedenti forniscono una base solida per riflettere in maniera critica sull'efficacia delle soluzioni proposte. In questa sezione si analizzeranno in dettaglio le evidenze emerse, mettendole in relazione con gli obiettivi della ricerca, con l'intento di offrire una visione complessiva e coerente delle performance ottenute.

6.1. Riconoscimento dei segni isolati

A seguito di un'ampia esplorazione dello spazio degli iperparametri, che ha coinvolto sia le architetture dei modelli sia le funzioni di perdita, è stato possibile delineare un quadro completo delle prestazioni raggiunte. L'analisi comparativa ha evidenziato la superiorità degli approcci di *deep learning* rispetto ai metodi classici di *machine learning*, con miglioramenti significativi in termini di accuratezza su entrambi i dataset considerati.

Come previsto, i modelli addestrati e testati sul dataset a 72 classi mostrano una lieve riduzione della qualità delle predizioni. Tale calo è attribuibile alla maggiore complessità del problema, caratterizzato da un numero più elevato di classi, una maggiore variabilità intra-classe e una separabilità ridotta tra segni visivamente simili.

Il modello che ha fornito le migliori prestazioni è risultato essere la rete **BiLSTM** addestrata con una funzione di perdita mista, composta da *categorical cross entropy* e *Triplet Loss*. Questa configurazione ha raggiunto un'accuratezza pari al **96.95%** sul dataset a 46 classi e al **92.9%** su quello a 72 classi. A distanza molto ravvicinata si colloca il modello **Transformer**, che ha ottenuto un'accuratezza del 96.4% e del 92.0% rispettivamente, nelle medesime condizioni di addestramento.

L'analisi complessiva dei risultati consente di trarre alcune considerazioni chiave sull'efficacia dei modelli impiegati per la classificazione dei segni singoli della LIS. In primo luogo, i modelli di *machine learning* classici hanno fornito una solida *baseline*, con le *Random Forest* che hanno raggiunto le migliori performance tra le soluzioni tradizionali. Tuttavia, la maggiore capacità rappresentativa dei modelli di *deep learning* ha permesso di ottenere miglioramenti significativi in termini di accuratezza, precisione e F1-score,

confermando l'importanza di approcci in grado di catturare la complessità sia temporale sia spaziale dei dati.

Tra le architetture di *deep learning*, le reti **LSTM** si sono distinte come soluzione particolarmente efficace, riuscendo a bilanciare elevata accuratezza e robustezza nella generalizzazione, soprattutto per il dataset a 72 classi. L'introduzione di funzioni di perdita aggiuntive, come la *Siamese Loss* e la *Triplet Loss*, ha ulteriormente migliorato le prestazioni, riducendo le ambiguità tra classi simili e favorendo una migliore separazione nello spazio degli *embedding*. In particolare, la combinazione della *categorical cross entropy* con la *Triplet Loss*, applicata alla rete LSTM, ha fornito i risultati più promettenti, raggiungendo un'accuratezza del **96.95%** sul dataset a 46 classi e del **92.9%** sul dataset a 72 classi.

Un aspetto rilevante è che le prestazioni di LSTM, 1D-CNN e Transformer risultano complessivamente comparabili. Sebbene la modellazione della componente temporale, tipica delle LSTM, garantisca un vantaggio marginale nei segni più complessi o visivamente simili, le differenze tra le architetture sono nel complesso contenute. Le matrici di confusione e l'analisi degli *embedding* confermano l'efficacia di tutte e tre le soluzioni nel ridurre gli errori più frequenti e nel separare correttamente le classi maggiormente problematiche.

In conclusione, i risultati suggeriscono che la combinazione di una rete **LSTM** con *categorical cross entropy* e *Triplet Loss* rappresenti la soluzione più adatta per il task di classificazione dei segni singoli della LIS. Tale approccio garantisce un bilanciamento ottimale tra accuratezza, capacità di generalizzazione e gestione delle ambiguità tra classi simili, offrendo una base solida per l'integrazione in futuri sistemi di riconoscimento in tempo reale.

6.2. Rilevamento Out-Of-Distribution

L'analisi dei risultati relativi al rilevamento dei segni *out-of-distribution* (OOD) permette di trarre alcune considerazioni sugli approcci valutati.

Il primo metodo, basato sulla combinazione di *feature extraction* tramite CNN e algoritmi classici di *anomaly detection*, ha fornito una *baseline* di riferimento, con un'accuratezza massima del 67% ottenuta mediante **Isolation Forest**.

L'approccio *logit-based* ha evidenziato come la rete **LSTM** costituisca la scelta più efficace tra le architetture considerate, con l'analisi dell'**entropia** che si è rivelata leggermente superiore rispetto al semplice *Max Probability* e alla funzione di energia, sia in termini di AUROC che di accuratezza OOD.

Infine, l'approccio *teacher-student* ha ottenuto le migliori prestazioni complessive in termini di accuratezza OOD, con l'architettura **Transformer** che ha raggiunto il valore massimo del **78.5%**.

Tuttavia, è importante sottolineare che nessuno dei metodi analizzati ha fornito una separazione netta tra segni *in-distribution* e *out-of-distribution*. Questo limite è probabilmente imputabile alla forte similarità tra i segni OOD e quelli IID, che ostacola una chiara distinzione nello spazio delle rappresentazioni e riduce l'efficacia dei metodi di rilevamento impiegati.

Tale limitazione non è legata a un bias del dataset, ma riflette una caratteristica intrinseca della lingua dei segni: segni che differiscono soltanto per pochi tratti possono assumere significati completamente diversi. Di conseguenza, anche quando un segno non è presente nel training set, la rete tende a considerarlo simile a quelli già osservati, enfatizzando le somiglianze, spesso prevalenti rispetto alle differenze. Una possibile estensione futura potrebbe consistere nell'integrazione di tecniche di *representation learning* più sofisticate, volte a migliorare la discriminabilità tra segni OOD e IID.

6.3. Classificazione Multisegno

La classificazione multisegno ha rappresentato la fase più complessa dell'intero progetto, poiché finalizzata alla traduzione di sequenze continue di segni della LIS nelle corrispondenti unità testuali. Rispetto alla classificazione di segni isolati, questo compito introduce sfide aggiuntive quali l'elevata variabilità dei movimenti, la gestione delle transizioni tra segni consecutivi e la presenza di rumore nei dati acquisiti.

L'analisi comparativa dei tre approcci proposti evidenzia un'evoluzione progressiva nelle prestazioni, in linea con le aspettative progettuali. L'approccio iniziale, basato su finestre temporali fisse, ha mostrato performance limitate, con un'elevata frequenza di errori sia nella classificazione dei singoli segni sia nella segmentazione delle sequenze. Questo risultato conferma il ruolo esplorativo del metodo, utile principalmente per evidenziare le criticità intrinseche del problema e motivare la necessità di soluzioni più avanzate.

La seconda strategia, che ha introdotto finestre temporali ridotte combinate con un meccanismo di *major voting*, ha ottenuto i migliori risultati complessivi. La riduzione della discrepanza tra dati di training e dati di test ha consentito una maggiore coerenza nell'elaborazione delle sequenze, migliorando sensibilmente l'accuratezza della classificazione e riducendo gli errori di segmentazione, in particolare per sequenze brevi come quelle contenenti lettere dell'alfabeto.

L'ultimo approccio, basato su un modello BiLSTM arricchito con meccanismi di attenzione e addestrato su un dataset sintetico, ha mostrato buone prestazioni sul dataset artificiale, ma con un leggero calo sui dati reali. Tale differenza è attribuibile alla complessità aggiuntiva dei video reali, caratterizzati da rumore e maggiore variabilità, che il modello fatica a gestire rispetto ai dati generati artificialmente.

Complessivamente, i risultati dimostrano come la progressiva sofisticazione degli approcci, dalla semplice segmentazione a finestre fisse fino alla strategia con finestre ridotte e *major voting*, conduca a un miglioramento significativo delle metriche di valutazione, tra cui *precision*, *recall*, *F1-score* e *WER*. Dal punto di vista applicativo, il secondo approccio rappresenta quindi la soluzione più promettente per futuri sistemi di traduzione automatica della LIS, in quanto coniuga accuratezza, robustezza e capacità di generalizzazione anche in presenza di variabilità temporale e rumore.

7 | Conclusioni e Sviluppi Futuri

Questo lavoro ha dimostrato come l'impiego di metodologie di Machine Learning e Deep Learning consenta lo sviluppo di sistemi in grado di riconoscere automaticamente i segni della Lingua dei Segni Italiana (LIS), sia nel contesto dei segni isolati sia in quello più complesso delle sequenze continue.

L'analisi è stata condotta su due dataset privati forniti dall'azienda Cludia Research, comprendenti un totale di 9.700 video suddivisi tra lettere, numeri e parole di uso quotidiano, con fino a 72 segni distinti e oltre 100 campioni per classe nel *dataset* di dimensioni maggiori. Questa disponibilità di dati ha contribuito a colmare una delle principali criticità di questo ambito di ricerca, ossia la scarsità e l'inadeguatezza dei *dataset* esistenti. Ha inoltre reso possibile una valutazione sistematica delle diverse architetture, fornendo una base solida per il confronto tra approcci eterogenei in condizioni sperimentali più vicine a scenari d'uso reali. Nel corso dello studio sono stati analizzati e sperimentati diversi approcci, a partire da modelli di Machine Learning tradizionale fino ad architetture di Deep Learning più avanzate, quali CNN, LSTM e Transformers. Ciascuna architettura è stata valutata in relazione alla capacità di modellare le componenti spaziali e temporali del segnale, con l'ulteriore estensione all'analisi dell'Out-of-Distribution Detection, mediante tecniche di anomaly detection, metodi logit-based e approcci teacher-student. Per il riconoscimento di sequenze continue, infine, sono stati confrontati un approccio basato su *sliding window* con *majority voting* e un modello end-to-end basato su LSTM, al fine di affrontare la complessità introdotta dai segni concatenati. I risultati ottenuti evidenziano come tali tecnologie possano costituire la base per la realizzazione di strumenti innovativi volti a ridurre in maniera significativa le barriere comunicative tra le persone sorde e la popolazione udente, favorendo un'interazione più naturale e bidirezionale.

Alla luce di questi progressi, questo studio apre la strada a numerosi sviluppi futuri. Un primo ambito di miglioramento riguarda l'espansione del dataset di segni isolati: quello attualmente impiegato, pur includendo 72 classi, è costituito prevalentemente da simboli, numeri e articolazioni, con una copertura limitata di parole di uso quotidiano. Tale caratteristica rende complesso costruire frasi articolate e concetti espressivi, limitando di

fatto la possibilità di sperimentare approcci di traduzione continua su sequenze linguisticamente più ricche. L'ampliamento del vocabolario rappresenterebbe quindi un passo cruciale per sviluppare modelli in grado di affrontare scenari comunicativi più realistici e vicini all'uso quotidiano della LIS. Parallelamente all'espansione del *dataset* di segni isolati, sarebbe altrettanto importante migliorare l'insieme di dati utilizzato in fase di test per la rilevazione e la classificazione dei segni continui. L'adozione di *dataset* più ampi e diversificati permetterebbe infatti di ottenere misure di prestazione più affidabili e generalizzabili, avvicinando le condizioni sperimentali a scenari d'uso reali e riducendo il rischio di *overfitting* sui dati attualmente disponibili.

Riguardo ai metodi utilizzati, una possibile direzione di ricerca riguarda il confronto tra l'approccio adottato in questo lavoro e le soluzioni comunemente impiegate in letteratura che non prevedono la fase di estrazione manuale delle feature. Rimanendo nell'ambito degli approcci skeleton-based, potrebbero essere esplorate architetture più avanzate come le Graph Convolutional Network (GCN), in grado di modellare in maniera più naturale la struttura non euclidea dei dati scheletrici. Inoltre, potrebbero essere valutati sistemi di fusione multimodale delle feature, che combinino le informazioni derivate dallo skeleton con quelle provenienti dal canale RGB, al fine di sfruttare in maniera complementare sia la componente spaziale sia quella semantica del segnale visivo. Un'ulteriore direzione di sviluppo, finalizzata a migliorare la qualità dell'output mediante una manipolazione più sofisticata della sequenza predetta, consiste nell'introduzione di filtri basati sulle regole grammaticali della LIS. Ad esempio, se viene rilevato un segno corrispondente a un verbo, il filtro potrebbe escludere la possibilità che il segno successivo sia nuovamente un verbo, garantendo una maggiore coerenza sintattica nella sequenza finale. Per quanto riguarda la classificazione delle singole lettere, al fine di renderla più robusta, sarebbe possibile integrare un Hidden Markov Model (HMM). In questo contesto, anziché limitarsi alla sola lettera con probabilità massima, si potrebbero considerare le tre lettere più probabili predette dal modello di base e utilizzare le probabilità condizionate sulle lettere precedentemente riconosciute per determinare la sequenza alfabetica più plausibile.

In conclusione, l'implementazione e la diffusione di sistemi di traduzione automatica della LIS potrebbero avere un impatto sociale di straordinaria portata, favorendo l'accessibilità e l'inclusione in ambiti fondamentali come l'istruzione, i servizi pubblici, la vita professionale e culturale. L'integrazione futura di modelli in grado di operare in tempo reale, su larga scala e in scenari non supervisionati rappresenterebbe un passo decisivo verso la realizzazione di interfacce multimodali capaci di garantire una comunicazione più inclusiva ed accessibile.

Bibliografia

- [1] Testo coordinato del decreto-legge 22 marzo 2021, n. 41, 2021. URL <https://www.gazzettaufficiale.it/eli/id/2021/05/21/21A03181/sg>. Gazzetta Ufficiale Serie Generale n.120 del 21-05-2021 - Suppl. Ordinario n. 21.
- [2] S. Alyami and H. Luqman. A comparative study of continuous sign language recognition techniques. *arXiv preprint arXiv:2406.12369*, 2024.
- [3] J. Alzubi, A. Nayyar, and A. Kumar. Machine learning from theory to algorithms: An overview. *Journal of Physics: Conference Series*, 1142(1):012012, nov 2018. doi: 10.1088/1742-6596/1142/1/012012. URL <https://dx.doi.org/10.1088/1742-6596/1142/1/012012>.
- [4] Aristotele. Storia degli animali. In D. Lanza and M. Vegetti, editors, *Opere biologiche*. UTET, Torino, 1971. Il passo di interesse si trova nel Libro IV, 9, 536b.
- [5] Aristotele. La sensazione e i sensibili (de sensu et sensibilibus). In R. Laurenti, editor, *Dell'anima. Piccoli trattati di storia naturale*. Laterza, Roma-Bari, 2005. Il passo fondamentale sul ruolo dell'udito per l'intelligenza si trova in 1, 437a.
- [6] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*, 2015.
- [7] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. In *IEEE transactions on neural networks*, pages 157–166. IEEE, 1994.
- [8] C. Branchini and L. Mantovan, editors. *Grammatica della lingua dei segni italiana (LIS)*. Edizioni Ca' Foscari, Venezia, 2022. ISBN 978-88-6969-645-9. URL <https://edizionicafoscari.unive.it/it/edizioni4/libri/978-88-6969-645-9/>. A cura di Chiara Branchini e Lara Mantovan.

- [9] A. Brettmann, J. Gravinghoff, M. Rüschoff, and M. Westhues. Breaking the barriers: Video vision transformers for word-level sign language recognition, 2025. URL <https://arxiv.org/abs/2504.07792>.
- [10] I. Bulugu. Tanzanian sign language recognition system for an assistive communication glove sign tutor based on the inertial sensor fusion control algorithm. *Journal of Electrical Systems and Information Technology*, 12(1):8, Mar. 2025. ISSN 2314-7172. doi: 10.1186/s43067-025-00199-9. URL <https://doi.org/10.1186/s43067-025-00199-9>.
- [11] G. Caligiore, R. Mineo, C. Spampinato, E. Ragonese, S. Palazzo, and S. Fontana. Multisource approaches to italian sign language (LIS) recognition: Insights from the multimedalis dataset. In F. DellOrletta, A. Lenci, S. Montemagni, and R. Sprugnoli, editors, *Proceedings of the 10th Italian Conference on Computational Linguistics (CLiC-it 2024)*, pages 132–140, Pisa, Italy, Dec. 2024. CEUR Workshop Proceedings. ISBN 979-12-210-7060-6. URL <https://aclanthology.org/2024.clicit-1.17/>.
- [12] C. Cecchetto, S. Giudice, and E. Mereghetti. La raccolta del corpus lis. In A. Cardinaletti, C. Cecchetto, and C. Donati, editors, *Grammatica, Lessico e Dimensioni di Variazione della LIS*, pages 55–68. FrancoAngeli, Milan, 2011.
- [13] J. L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.
- [14] Ente Nazionale Sordi (ENS). Lingua dei segni italiana (lis), 2025. URL <https://www.ens.it/lingua-dei-segni/>. Pagina ufficiale sull’Italian Sign Language (LIS).
- [15] M. Erard. Why sign-language gloves don’t help deaf people, 2017. URL <https://www.theatlantic.com/technology/archive/2017/11/why-sign-language-gloves-dont-help-deaf-people/545441/>.
- [16] K. Ertürk, F. Altınışık, İrem Sarialetn, and Ömer Nezih Gerek. Tslformer: A lightweight transformer model for turkish sign language recognition using skeletal landmarks, 2025. URL <https://arxiv.org/abs/2505.07890>.
- [17] M. Fagiani, S. Squartini, E. Principi, and F. Piazza. A new italian sign language database. 07 2012. ISBN 978-3-642-31560-2. doi: 10.1007/978-3-642-31561-9_18.
- [18] A. Filipowska, W. Filipowski, P. Raif, M. Pieniążek, J. Bodak, P. Ferst, K. Pilarski, S. Sieciński, R. J. Doniec, J. Mieszczanin, E. Skwarek, K. Bryzik, M. Henkel, and M. Grzegorzec. Machine learning-based gesture recognition glove: Design and implementation. *Sensors*, 24(18), 2024. ISSN 1424-8220. doi: 10.3390/s24186157. URL <https://www.mdpi.com/1424-8220/24/18/6157>.

- [19] J. Forster, C. Schmidt, O. Koller, M. Bellgardt, and H. Ney. Extensions of the sign language recognition and translation corpus rwth-phoenix-weather. volume 1, 05 2014.
- [20] S.-W. Gan, Y.-F. Yin, Z.-W. Jiang, L. Xie, and S.-L. Lu. Vision-based sign language translation via a skeleton-aware neural network. *Journal of Computer Science and Technology*, 40(2):378–396, 2025. doi: 10.1007/s11390-024-2978-y. URL <https://doi.org/10.1007/s11390-024-2978-y>.
- [21] F. A. Gers, J. Schmidhuber, and F. Cummins. Learning to forget: Continual prediction with lstm. In *Neural Computation*, pages 2451–2471. MIT Press, 2000.
- [22] Giustiniano. *Le Istituzioni dell’Imperatore Giustiniano. Libri quattro*. Vecchioni & Figli, L’Aquila / Italia, 1908. Cfr. Libro I, Titolo XXIII, §4.
- [23] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.
- [24] W. He, F. Wan, Y. Liu, G. Wu, P. Zhang, Y. Xiong, X. Zhang, T. Hang, W. Chen, K. Chen, B. Xue, R. Li, G. Hu, Z. Li, Y. Wu, J. Zhu, T. Xiang, J. Zheng, and Y. Zhang. Deep learning-powered composite foam smart glove with cross-dimensional conductive networks for sign language recognition. *Composites Part B: Engineering*, 308:112985, 2026. ISSN 1359-8368. doi: <https://doi.org/10.1016/j.compositesb.2025.112985>. URL <https://www.sciencedirect.com/science/article/pii/S1359836825008960>.
- [25] M. Hilzensauer and K. Krammer. A multilingual dictionary for sign languages: "spreadthesign". 11 2015.
- [26] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735.
- [27] L. Hu, L. Gao, Z. Liu, and W. Feng. Dynamic spatial-temporal aggregation for skeleton-aware sign language recognition, 2024. URL <https://arxiv.org/abs/2403.12519>.
- [28] J. Huang and V. Chouvatut. Video-based sign language recognition via resnet and lstm network. *Journal of Imaging*, 10(6), 2024. ISSN 2313-433X. doi: 10.3390/jimaging10060149. URL <https://www.mdpi.com/2313-433X/10/6/149>.
- [29] A. Ji, Y. Wang, X. Miao, T. Fan, B. Ru, L. Liu, R. Nie, and S. Qiu. Dataglove for sign language recognition of people with hearing and speech impairment via wearable inertial sensors. *Sensors (Basel)*, 23(15):6693, July 2023. doi: 10.3390/s23156693.

- [30] S. Jiang, B. Sun, L. Wang, Y. Bai, K. Li, and Y. Fu. Skeleton aware multi-modal sign language recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2021.
- [31] H. R. V. Joze and O. Koller. Ms-asl: A large-scale data set and benchmark for understanding american sign language, 2019. URL <https://arxiv.org/abs/1812.01053>.
- [32] T. Kapuściński and D. Warchoń. Hand posture recognition using skeletal data and distance descriptor. *Applied Sciences*, 10(6), 2020. ISSN 2076-3417. doi: 10.3390/app10062132. URL <https://www.mdpi.com/2076-3417/10/6/2132>.
- [33] H. Kim, I. Lee, S. Panda, S. Hajra, B. Jeong, J. Seo, K. R. Kaja, M. A. Belal, V. Vivekananthan, and H. J. Kim. Smart gloves-based triboelectric nanogenerator for sign language detection. *Micro and Nano Systems Letters*, 13(1):11, July 2025. ISSN 2213-9621. doi: 10.1186/s40486-025-00231-7. URL <https://doi.org/10.1186/s40486-025-00231-7>.
- [34] J. Kramer and L. Leifer. The talking glove. *SIGCAPH Comput. Phys. Handicap.*, (39):12–16, Apr. 1988. ISSN 0163-5727. doi: 10.1145/47937.47938. URL <https://doi.org/10.1145/47937.47938>.
- [35] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.
- [36] D. Kumari and R. Anand. Isolated video-based sign language recognition using a hybrid cnn-lstm framework based on attention mechanism. *Electronics*, 13:1229, 03 2024. doi: 10.3390/electronics13071229.
- [37] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.
- [38] D. Li, C. Rodriguez, X. Yu, and H. Li. Word-level deep sign language recognition from video: A new large-scale dataset and methods comparison. In *The IEEE Winter Conference on Applications of Computer Vision*, pages 1459–1469, 2020.
- [39] Z. Long, X. Liu, J. Qiao, and Z. Li. Sign language recognition based on facial expression and hand skeleton. In *2023 38th Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, page 237–241. IEEE, Aug. 2023. doi: 10.1109/yac59482.2023.10401630. URL <http://dx.doi.org/10.1109/YAC59482.2023.10401630>.

- [40] C. Lugaresi, J. Tang, H. Nash, C. McClanahan, E. Uboweja, M. Hays, F. Zhang, C.-L. Chang, M. G. Yong, J. Lee, W.-T. Chang, W. Hua, M. Georg, and M. Grundmann. Mediapipe: A framework for building perception pipelines, 2019. URL <https://arxiv.org/abs/1906.08172>.
- [41] M.-T. Luong, H. Pham, and C. D. Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, 2015.
- [42] M. Madhiarasan and P. P. Roy. A comprehensive review of sign language recognition: Different types, modalities, and datasets. *arXiv preprint arXiv:2204.03328*, 2022. URL <https://arxiv.org/pdf/2204.03328>.
- [43] B. Mahesh. Machine learning algorithms -a review, 01 2019.
- [44] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.
- [45] A. S. M. Miah, M. A. M. Hasan, S.-W. Jang, H.-S. Lee, and J. Shin. Multi-stream general and graph-based deep neural networks for skeleton-based sign language recognition. *Electronics*, 12(13), 2023. ISSN 2079-9292. doi: 10.3390/electronics12132841. URL <https://www.mdpi.com/2079-9292/12/13/2841>.
- [46] A. S. M. Miah, M. A. M. Hasan, S. Nishimura, and J. Shin. Sign language recognition using graph and general deep neural network based on large scale dataset. *IEEE Access*, 12:34553–34569, 2024. doi: 10.1109/ACCESS.2024.3372425.
- [47] R. Mineo, G. Caligiore, C. Spampinato, S. Fontana, S. Palazzo, and E. Ragonese. Sign language recognition for patient-doctor communication: A multimedia/multimodal dataset. In *2024 IEEE 8th Forum on Research and Technologies for Society and Industry Innovation (RTSI)*, pages 202–207. IEEE, 2024.
- [48] T. M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997. ISBN 978-0070428072.
- [49] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML)*, pages 807–814, Haifa, Israel, 2010. Omnipress. ICML 2010.
- [50] K. P. Nimisha and A. Jacob. A brief review of the recent trends in sign language recognition. 2020. URL <https://ieeexplore.ieee.org/document/9182351>.

- [51] S. Patra, A. Maitra, M. Tiwari, K. Kumaran, S. Prabhu, S. Punyeshwarananda, and S. Samanta. Hierarchical windowed graph attention network and a large scale dataset for isolated indian sign language recognition, 2024. URL <https://arxiv.org/abs/2407.14224>.
- [52] F. Pezzuoli, D. Corona, and M. L. Corradini. Dynamic gestures recognition through a low-cost data glove. In *2020 IEEE International Conference on Human-Machine Systems (ICHMS)*, pages 1–3, 2020. doi: 10.1109/ICHMS49158.2020.9209424.
- [53] L. Pigou, M. Van Herreweghe, and J. Dambre. Sign language recognition using convolutional neural networks. In *Computer Vision – ECCV 2014 Workshops*, pages 572–578. Springer, 2015. ISBN 9783319161785.
- [54] Platone. Cratilo. In G. Reale, editor, *Tutti gli scritti*. Bompiani, Milano, 2000. Il passo di interesse si trova in 422e-423b.
- [55] T. Pryor and N. Azodi. Signaloud: Gloves that transliterate sign language into text and speech. Lemelson-MIT “Use it!” Undergraduate Winners Fact Sheet, 2020. URL https://lemelson.mit.edu/sites/default/files/2020-09/Signaloud_Fact_Sheet.pdf. \$10,000 Lemelson-MIT “Use it!” Undergraduate Winners.
- [56] F. Ronchetti, F. Quiroga, C. Estrebou, L. Lanzarini, and A. Rosete. Lsa64: A dataset of argentinian sign language. In *XX II Congreso Argentino de Ciencias de la Computación (CACIC)*, volume 1, page 3, 2016.
- [57] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.
- [58] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.
- [59] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.
- [60] A. L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):210–229, 1959.
- [61] F. d. Saussure. *Course in General Linguistics*. Columbia University Press, New York, 2011. ISBN 9780231157261.
- [62] J. Shin, M. A. M. Hasan, A. S. M. Miah, K. Suzuki, and K. Hirooka. Japanese sign language recognition by combining joint skeleton-based handcrafted and pixel-based

- deep learning features with machine learning classification. *Comput. Model. Eng. Sci.*, 139(3):2605–2625, 2024.
- [63] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [64] SpreadTheSign. Spreadthesign - online sign language dictionary. <https://www.spreadthesign.com>. Accessed: 2025-09-17.
- [65] T. Starner and A. Pentland. Real-time american sign language recognition from video using hidden markov models. In *Proceedings of International Symposium on Computer Vision - ISCV*, pages 265–270, 1995. doi: 10.1109/ISCV.1995.477012.
- [66] K. Sun, B. Xiao, D. Liu, and J. Wang. Deep high-resolution representation learning for human pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5693–5703, Long Beach, CA, USA, June 2019.
- [67] C. Szegedy et al. Going deeper with convolutions. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.
- [68] S. Tamura and S. Kawasaki. Recognition of sign language motion images. *Pattern Recognition*, 21(4):343–353, 1988. ISSN 0031-3203. doi: [https://doi.org/10.1016/0031-3203\(88\)90048-9](https://doi.org/10.1016/0031-3203(88)90048-9). URL <https://www.sciencedirect.com/science/article/pii/0031320388900489>.
- [69] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017. URL <https://doi.org/10.48550/arXiv.1706.03762>.
- [70] V. Volterra. 6 lezioni sulla lis, n.d. URL <https://www.youtube.com/playlist?list=PLWA0dsP-DBGDhe8o0nTXHtX5zStCQ4rKI>. Università Telematica Internazionale UNINETTUNO.
- [71] U. von Agris, M. Knorr, and K.-F. Kraiss. The significance of facial features for automatic sign language recognition. In *2008 8th IEEE International Conference on Automatic Face Gesture Recognition*, pages 1–6, 2008. doi: 10.1109/AFGR.2008.4813472.
- [72] D. Wei, H. Hu, and G.-F. Ma. Part-wise graph fourier learning for skeleton-based continuous sign language recognition. *Journal of Imaging*, 11(8), 2025. ISSN 2313-433X. doi: 10.3390/jimaging11080286. URL <https://www.mdpi.com/2313-433X/11/8/286>.

- [73] World Health Organization. Deafness and hearing loss, 2025. URL <https://www.who.int/news-room/fact-sheets/detail/deafness-and-hearing-loss>. Accessed: 2025-07-24.
- [74] Q. Zhou, H. Li, W. Meng, H. Dai, T. Zhou, and G. Zheng. Fusion of multimodal spatio-temporal features and 3d deformable convolution based on sign language recognition in sensor networks. *Sensors*, 25(14), 2025. ISSN 1424-8220. doi: 10.3390/s25144378. URL <https://www.mdpi.com/1424-8220/25/14/4378>.
- [75] Z. Zhou, K. Chen, X. Li, S. Zhang, Y. Wu, Y. Zhou, K. Meng, C. Sun, Q. He, W. Fan, E. Fan, Z. Lin, X. Tan, W. Deng, J. Yang, and J. Chen. Sign-to-speech translation using machine-learning-assisted stretchable sensor arrays. *Nature Electronics*, 3(9): 571–578, 2020. doi: 10.1038/s41928-020-0428-6. URL <https://doi.org/10.1038/s41928-020-0428-6>.

Elenco delle figure

2.1	Segno "Lavoro"	10
2.2	Segno "Prestito"	10
2.3	Segni "Lavoro" e "Prestito" della LIS. Illustrazioni realizzate dall'autore a partire da materiali presenti in [64].	10
2.4	Segno "Casa"	11
2.5	Segno "Telefono"	11
2.6	Segni "Casa" e "Telefono" della LIS. Illustrazioni realizzate dall'autore a partire da materiali presenti in [64].	11
2.7	Il segno per "cane" in tre lingue dei segni: LIS (Lingua dei Segni Italiana), ASL (American Sign Language) e BSL (British Sign Language). L'immagine evidenzia come ciascuna lingua performi il segno in maniera differente. Illustrazioni realizzate dall'autore a partire da materiali presenti in [64]. . .	12
2.8	Schema di una cella LSTM, con i tre gate principali: <i>input</i> , <i>forget</i> e <i>output</i> , e il flusso delle informazioni attraverso lo stato di cella e lo stato nascosto.	18
3.1	Struttura tipica per la traduzione automatica in tempo reale della lingua dei segni, articolato in fasi di acquisizione, elaborazione e classificazione del segnale gestuale.	24
3.2	Pipeline del processo di addestramento in un sistema di riconoscimento dei segni: dalla raccolta dei dati, passando per le fasi di preprocessamento e di estrazione delle caratteristiche, fino all'addestramento e alla valutazione del modello.	25
3.3	Rappresentazione schematica (a) e fotografia reale (b) della mano di un soggetto su cui è stato applicato il dispositivo YSSA, insieme a un trasmettitore wireless.[75]	27
4.1	Classi che compongono il dataste di 72 segni con i relative frequenze	38
4.2	Architettura di una rete neurale con CNN monodimensionale	40
4.3	Architettura di una rete neurale basato su Bi-LSTM e meccanismo di attenzione	42

4.4	Architettura di un modello basato su Transformer Encoder	44
5.1	Confusion Matrix (46 classi)	58
5.2	Training Loss (46 classi)	59
5.3	Confusion Matrix (72 classi)	59
5.4	Training Loss (72 classi)	60
5.5	Confronto tra le rappresentazioni nello spazio degli <i>embedding</i> generate da una 1D-CNN addestrata con due diverse funzioni di perdita.	61

Elenco delle tabelle

5.1	Prestazioni dei modelli sui due dataset (46 e 72 classi) in termini di Accuracy, Weighted Precision e Weighted F1-score.	56
5.2	Prestazioni dei modelli di Deep Learning addestrati con la <i>categorical cross entropy</i> sui due dataset (46 e 72 classi), riportate in termini di Accuracy, Weighted Precision e Weighted F1-score.	57
5.3	Prestazioni dei modelli di Deep Learning con diverse funzioni di loss sui due dataset (46 e 72 classi), in termini di Accuracy, Weighted Precision e Weighted F1-score.	62
5.4	Prestazioni in % dei metodi di anomaly detection applicati agli <i>embedding</i> estratti tramite CNN, in termini di AUROC, ACC e F1-score OOD.	64
5.5	Risultati OOD basati su <i>Max Probability</i> per le diverse architetture.	65
5.6	Risultati OOD basati su <i>Entropy</i> per le diverse architetture.	65
5.7	Risultati OOD basati su <i>Energy</i> per le diverse architetture.	65
5.8	Confronto tra modelli Teacher e Student con differenti strategie di addestramento.	66
5.9	Risultati di rilevazione OOD per le diverse architetture.	66
5.10	Risultati del metodo baseline basato su finestre fisse per la classificazione multisegno.	68
5.11	Risultati dei modelli sul dataset di testing dei segni isolati di lunghezza 15 frame	69
5.12	Risultati del metodo basato su finestre ridotte e Major Voting.	70
5.13	Risultati del migliore modello end-to-end sul dataset di testing	70

Ringraziamenti

Questo lavoro rappresenta la conclusione di un fantastico viaggio durato cinque anni; quest'esperienza non solo mi ha formato dal punto di vista accademico, ma mi ha anche permesso di crescere come persona, insegnandomi il valore dello studio, della passione e della dedizione.

Un sincero ringraziamento va alla mia relatrice, Prof.ssa Maristella Matera, e alla correlatrice, Ludovica Piro, per la guida costante, i preziosi consigli e il supporto durante l'intero percorso di tesi.

Desidero inoltre esprimere la mia gratitudine ai miei genitori, che mi hanno dato l'opportunità di intraprendere questo percorso, alla mia famiglia, sempre vicina anche nei momenti più difficili, e ai miei amici, sempre pronti a strapparmi un sorriso e a sostenermi quando ne avevo più bisogno.

