UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA IN INGEGNERIA INFORMATICA

"APPLICAZIONE DELLA COMUNICAZIONE SEMANTICA A DISPOSITIVI DI INPUT"

Relatore: Prof. Stefano Tomasin

Laureando: Marco Giacomin

ANNO ACCADEMICO 2022 - 2023

Data di laurea 25/9/2023

Abstract

Il nuovo paradigma della Comunicazione Semantica (*Semantic Communication, SC*) sta rivoluzionando il campo delle telecomunicazioni con sistemi che comprendono ed estraggono il significato dei messaggi, superando i limiti dei sistemi attuali. Questo progetto esplora l'applicazione di questi principi all'utilizzo di dispositivi di input, in particolare ai mouse. Il lavoro consiste nella raccolta di dati di movimento del mouse impartiti da un utente e nella loro elaborazione attraverso metodi di apprendimento automatico, supervisionato e non, con l'obiettivo di costruire un modello capace di prevedere in modo affidabile lo spostamento del puntatore a partire da una sottosequenza minima dei dati completi. Anche se il modello risultante non ha soddisfatto le aspettative, la ricerca ha rivelato promettenti metodi alternativi esplorabili da progetti futuri per ottenere una previsione affidabile.

Indice

Introduzione

Nei pochi anni dalla sua emersione, il nuovo paradigma della Comunicazione Semantica (*Semantic Communication* o *SC*) sta rivoluzionando il campo delle telecomunicazioni tramite la costruzione di sistemi e canali capaci di comprendere ed estrarre il significato dei messaggi in transito. Laddove un canale di comunicazione classicamente inteso vuole trasmettere il testo del messaggio con quanta più fedeltà possibile, i nuovi canali semantici si concentrano sul preservare il suo significato, valore che può essere estratto con varie tecniche di apprendimento automatico e che può risultare più leggero del testo originale, superando così i limiti teorici dell'efficienza di trasmissione.

Questo progetto mira ad applicare questi principi alla categoria di comunicazione uomo-macchina dei dispositivi di input, in particolare ai mouse.

Nel capitolo 1 discuteremo la definizione formale di SC e descriveremo l'applicazione creata per la raccolta dei dati.

Nel capitolo 2 descriveremo l'implementazione del primo modello di apprendimento, che avrà il compito di effettuare un raggruppamento preliminare dei dati, e la formattazione dei dati necessaria al suo addestramento.

Nel capitolo 3 descriveremo la struttura e la creazione del secondo modello, che tramite una rete neurale avrà il compito di prevedere il significato di una sequenza di dati dato il suo sottoinsieme iniziale di lunghezza minima.

Nel capitolo 4, con l'ausilio di visualizzazioni grafiche, esporremo i risultati dell'addestramento dei modelli, la conoscenza da loro appresa sul dominio dei dati in ingresso, e infine l'accuratezza delle loro previsioni

Nel capitolo 5, infine, esporremo le considerazioni finali del progetto e le prospettive di ricerca future aperte dagli ambiti in cui i modelli addestrati non si sono rivelati all'altezza delle aspettative

Nota: Per descrivere dati multidimensionali si fa talvolta ricorso alla notazione usata nella libreria python `numpy` in cui un tensore ha una certa *forma*, una tupla di numeri naturali che indica l'estensione in ogni direzione della matrice. Per esempio, la matrice:

```
[
    [1,2,3],
    [4,5,6]
]
```

ha forma `(2,3)`. Un array monodimensionale di lunghezza n ha forma `(n,)` (si noti la virgola terminale che denota una sequenza a un solo elemento).

Capitolo 1: Comunicazione Semantica

1.1: Definizione

In "*A Paradigm Shift toward Semantic Communications*"[1], Niu et al. propongono una distinzione fondamentale tra il livello tecnico di comunicazione (da loro chiamato *LEVEL-A/LIVELLO-A*) e il livello semantico (*LEVEL-B/LIVELLO-B*). La tesi fondamentale dell'equipe è la seguente: da decenni i sistemi di comunicazione avvengono soltanto al LIVELLO-A, che comporta grosse limitazioni e dispendi di dati ed energia, problemi che potrebbero essere risolti dall'adozione del LIVELLO-B come strato di compressione/prioritizzazione intelligente dei messaggi.

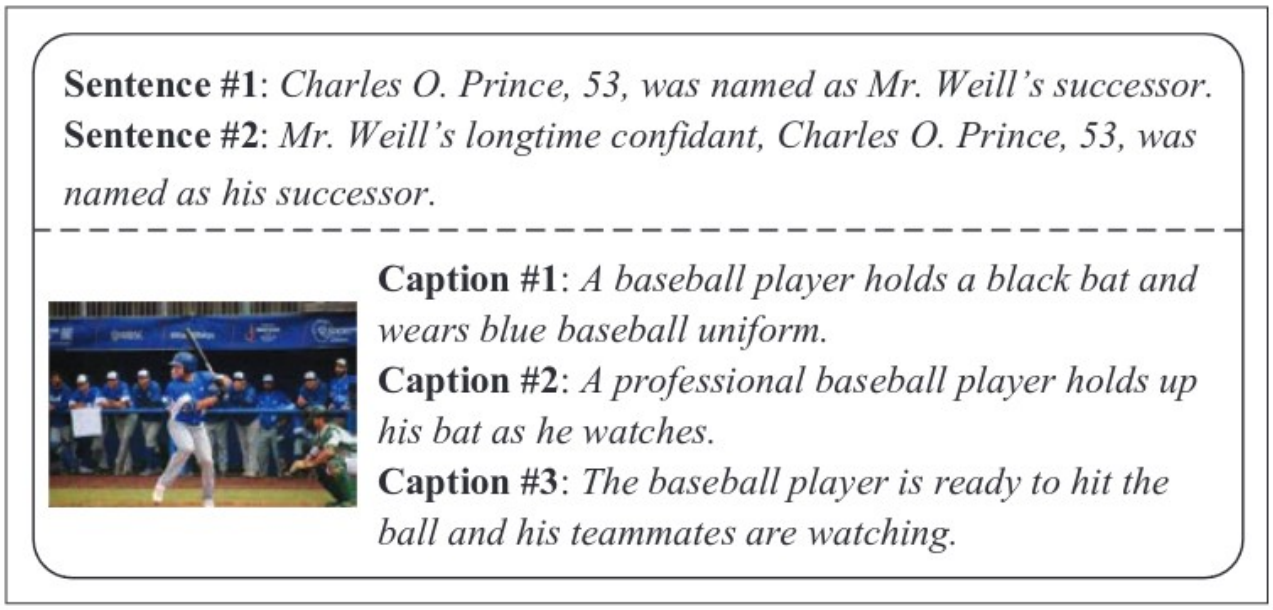

Figura 1: Esempio di espressioni con informazione semantica simile

Di seguito si riporta parte del testo del lavoro.

> "La figura 1 presenta due esempi. La coppia di frasi in cima costituisce delle parafrasi. Esse differiscono sul livello tecnico ma si sovrappongono su quello semantico. L'esempio in fondo elenca alcune descrizioni dell'immagine, che hanno tratti in comune pur mantenendo delle differenze. Ne segue che le semantiche del messaggio possono essere varie e gerarchizzate. La somiglianza del significato semantico per il trasmettitore e il ricevitore è ciò che importa nei sistemi di comunicazione del LIVELLO-B. Tuttavia, i problemi al LIVELLO-B richiedono ancora ricerche approfondite.
>
> Il sistema di comunicazione al LIVELLO-B dovrebbe presentare due caratteristiche principali.

1 K. Niu et al., "A Paradigm Shift toward Semantic Communications", IEEE Commun. Mag., Nov. 2022

Innanzitutto, questa nuova comunicazione semantica è istruita da conoscenze pregresse, il che copre la conoscenza riguardante le fonti e gli ambienti di trasmissione. Il contenuto sorgente prende parte al sistema di comunicazione, interagendo direttamente e agevolmente con altri moduli di trasmissione nel canale dotati di funzioni intelligenti. Le correlazioni semantiche di alto livello dentro ai dati sorgente verranno catturati invece di essere meramente impiegati nello studio delle proprietà statistiche. La semantica associata al linguaggio è stata inizialmente studiata nell'analisi linguistica. Inoltre, recenti innovazioni nelle tecnologie di IA hanno rivelato nuove intuizioni nel trattamento e codifica di informazioni relative alla semantica. Oltre ai dati sorgente, l'ambiente di canale wireless contiene informazioni semantiche. Il trasmettitore può selezionare le modalità più adatte per trasmettere l'informazione semantica di sorgente.

In secondo luogo, il sistema di comunicazione semantica è orientato agli incarichi (*task*). L'obiettivo definitivo è realizzare il sistema di comunicazione intelligente. Quindi, solo l'informazione necessaria rilevante all'incarico specifico al ricevente sarà trasmessa nel sistema di comunicazione semantica. Le parti irrilevanti verranno abbandonate o de-enfatizzate.

Questo cambiamento di metodologia nella comprensione completa dei dati sorgente e degli ambienti di trasmissione conduce finalmente al cambio di paradigma di trasmissione, dando forma al sistema di comunicazione semantica, come proposto nella sezione successiva.

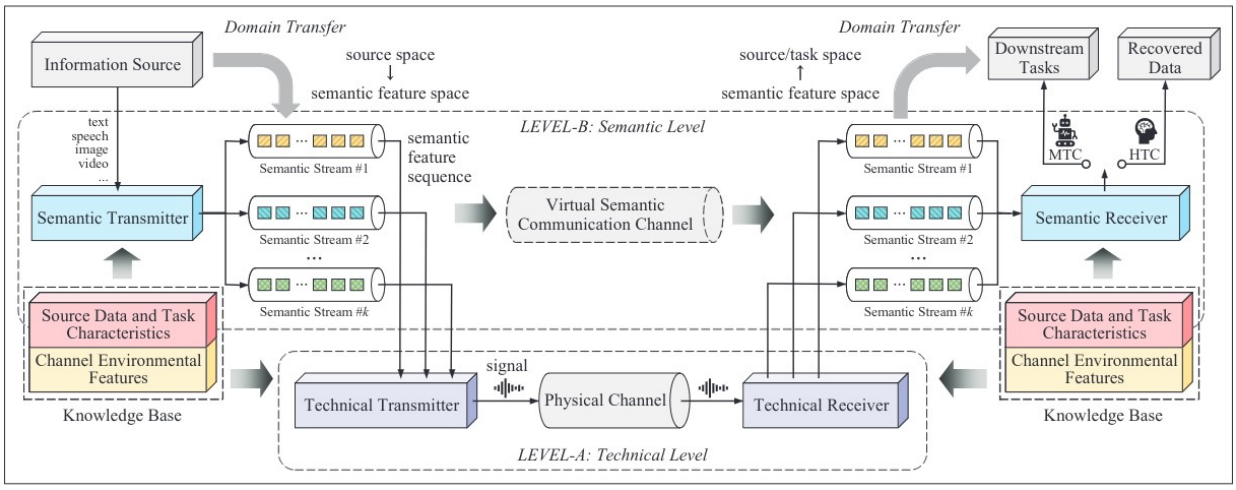

Figura 2: Schema di un sistema di comunicazione semantica

Sistemi di comunicazione semantica

Con "sistema di comunicazione semantica" intendiamo un sistema del tipo indicato schematicamente in fig. 2. Esso è una struttura gerarchica costituita sia dal livello tecnico che dal livello semantico. Nello specifico, il sistema include le seguenti parti:

- Una *fonte di informazioni* che produce dati da trasmettere alla destinazione. […] Dati di diverse modalità sono caratterizzati da diversi livelli di densità e ridondanza di informazioni

- Una *base di conoscenza* (*knowledge base/KB*) locale che fornisce ulteriore guida al transricevitore semantico. Questa base di conoscenza di sottofondo esiste in vari formati, tra cui *grafi di conoscenza*, *database*, *modelli addestrati* (parametrici o non). Questa fornisce anche conoscenze a priori sull'informazione dello stato del canale sia per il mittente che per il destinatario. Con la KB, il trasmettitore semantico ottiene un trattamento dei dati efficace, che enfatizza le porzioni importanti dei dati sorgente e si adegua alle caratteristiche del canale.

- Un *trasmettitore semantico* che opera sui dati sorgente per estrarre le caratteristiche relative alla semantica, che possono essere categorizzate in multipli flussi semantici, ognuno corrispondente a una caratteristica semantica dei dati sorgente. In aggiunta, ogni caratteristica semantica corrisponde a una differente importanza semantica per compiti specifici al ricevitore, incluso il recupero dati o compiti intelligenti in punti successivi del flusso, come traduzioni linguistiche o riconoscimento di oggetti. Le caratteristiche sono ulteriormente trattate e protette con priorità dettate dallo stato del canale.

- Il *trasmettitore tecnico*, *canale fisico* e *ricevitore tecnico* operano sulla sequenza di caratteristiche semantiche di ciascun flusso semantico in un modo che produce segnali adatti a essere trasmessi sul canale fisico. Possono utilizzare la codifica a separazione di sorgente e canale tradizionale combinata con la modulazione per produrre segnali digitali. Possono anche generare direttamente il segnale analogico usando la codifica unificata sorgente-canale emergente con delle reti neurali profonde. Il ricevitore tecnico esegue solitamente l'operazione inversa rispetto a quella fatta dal trasmettitore tecnico, ricostruendo il messaggio a partire dal segnale.

- Il *ricevitore semantico* esegue l'operazione inversa rispetto a quella fatta dal trasmettitore semantico, sfruttando la fusione semantica per ricostruire i dati sorgente o per eseguire direttamente dei compiti intelligenti più avanti nel flusso di informazione.

In sintesi, la caratteristica del sistema di comunicazione semantica proposto è il trattamento congiunto di sorgente e canale, con l'aiuto della base di conoscenza condivisa da trasmettitore e ricevitore."

1.2: Impiego dei principi di comunicazione semantica

L'adozione di sistemi di codifica di SC ha il potenziale di migliorare le prestazioni della comunicazione e ottimizzarle ai fini dei compiti che il sistema di destinazione deve svolgere. Vedremo in questo progetto un'applicazione specifica di questi sistemi per facilitare la trasmissione di una specifica classe di informazione (spostamento di un mouse) a uno specifico compito (l'interazione con un videogioco). Il modello che verrà descritto e implementato a seguire, pur non utilizzando la terminologia descritta dagli autori dell'articolo sopra citato, può essere considerato un esempio di trasmettitore semantico con base di conoscenza integrata.

1.3: Il gioco

Si consideri il seguente scenario: con l'ausilio di un mouse, un utente sta giocando a un videogioco il cui obiettivo è spostare ripetutamente un oggetto a schermo da una casella di partenza a una casella di destinazione. Raggiunta la casella di destinazione, ne viene immediatamente selezionata una nuova e comunicata visivamente all'utente, che dovrà cercare di farla raggiungere all'oggetto a partire dalla casella di destinazione precedente. Le caselle che l'oggetto può occupare sono in numero finito e visibili in ogni istante durante il gioco.

Chiamiamo la traslazione relativa complessiva dell'oggetto a seguito di un'interazione dell'utente uno **spostamento**. Dato che le caselle sono in numero finito, anche gli spostamenti tra esse saranno in numero finito di **classi**.

L'operazione di spostamento avviene per trascinamento. In ordine, l'utente:

- Sposta il puntatore sopra all'oggetto
- Preme e tiene premuto il tasto sinistro del mouse
- Muovendo fisicamente il mouse, sposta l'oggetto nel mondo di gioco verso la casella desiderata
- Una volta raggiunta la casella desiderata, rilascia il tasto sinistro del mouse

Le operazioni di spostamento sono gli unici istanti in cui l'oggetto non si trova in una delle caselle, al momento del rilascio del tasto sinistro del mouse l'oggetto viene immediatamente riposizionato dal gioco sulla casella più vicina (dove la misura della distanza è scelta in base alla configurazione delle caselle, per esempio distanza di Manhattan nel caso di una griglia quadrata regolare).

L'unico obiettivo del gioco è spostare l'oggetto sulla casella di destinazione, non ci sono requisiti sulla precisa strada che l'oggetto deve percorrere e il gioco può essere ripetuto un numero arbitrario di volte a discrezione dell'utente. A seconda della configurazione delle caselle, questa descrizione vale per una vasta classe di giochi, dagli scacchi al gioco dell'oca.

1.4: Applicazione della comunicazione semantica allo scenario

Nello scenario appena descritto l'informazione più importante che deve essere trasmessa dal mouse al gioco è la classe a cui appartiene lo spostamento da far compiere all'oggetto, ovvero la posizione relativa tra la casella di destinazione e quella di partenza. Tuttavia, la comunicazione convenzionale comporta l'invio di molti più bit di quanto strettamente necessario per via della necessità di descrivere istante per istante il movimento preciso del mouse dal momento in cui prende l'oggetto al

momento in cui lo rilascia, anziché lo spostamento dalla casella di partenza a quello di destinazione. Secondo i principi della comunicazione semantica, è necessario quindi introdurre a monte della comunicazione un modello capace di comprendere ed estrarre il suo significato complessivo (in questo caso lo spostamento desiderato dall'utente).

Obiettivo di questo progetto di tesi è la creazione di questo modello. Il suo compito sarà quello di apprendere gli spostamenti possibili (e, di conseguenza, forma e locazione delle posizioni valide) e di effettuare una previsione dello spostamento finale dato il sottoinsieme minimo possibile dei dati in ingresso. Questo rappresenta un esempio di comunicazione semantica, poiché una versione sufficientemente performante del modello potrebbe essere integrata nell'hardware del mouse, consentendogli di estrarre il significato del messaggio e risparmiando l'invio di informazioni superflue.

1.5: Il modello

Il modello di apprendimento sarà articolato in due componenti. Il primo, prendendo come input la sequenza di dati di un movimento, ne estrarrà il significato (lo spostamento). Il secondo, utilizzando il primo durante l'addestramento per etichettare i dati in ingresso, effettuerà la stessa previsione a partire dal sottoinsieme iniziale di lunghezza minima del dato di movimento.

1.6: Applicazione per raccolta dati – Descrizione

Al fine di acquisire i dati necessari per l’addestramento del modello, viene realizzata un’applicazione grafica che implementa il gioco descritto nella sezione 1.1.

L'applicazione prende la forma di un videogioco in cui un utente umano deve utilizzare un dispositivo di input (mouse, touchscreen, controller, etc.) al fine di spostare un oggetto virtuale (esistente in uno spazio 2D visualizzato a schermo) dal suo punto di partenza a quello marcato visivamente come "bersaglio". Al suo raggiungimento, viene indicata una nuova destinazione selezionata casualmente e il gioco può essere così ripetuto per un numero arbitrario di volte.

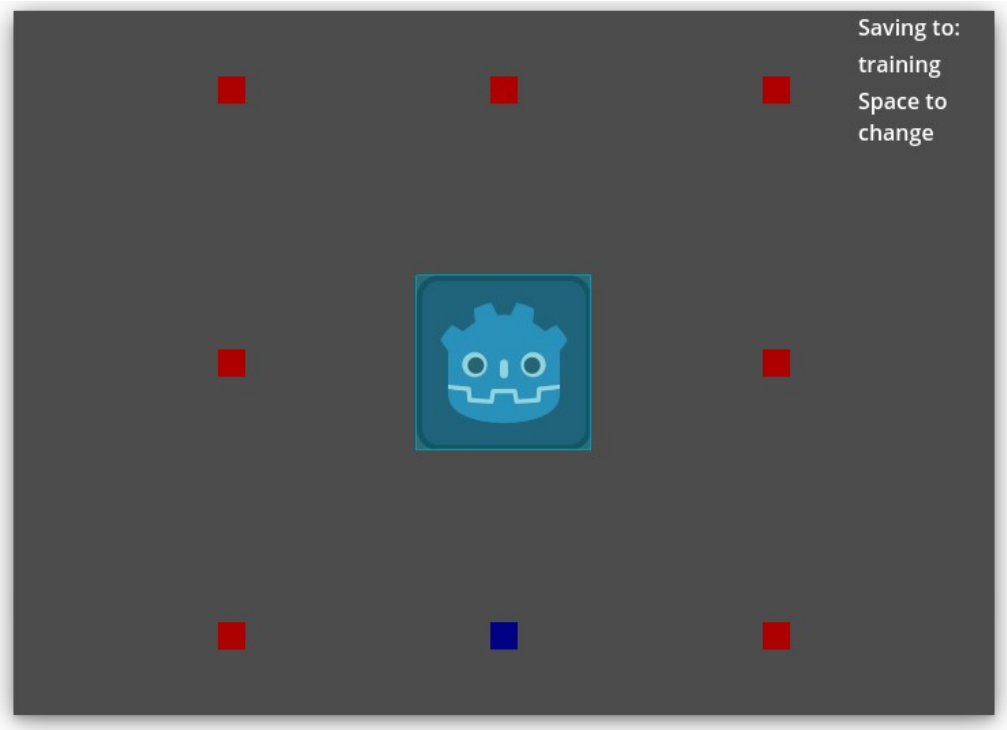

Figura 3: Schermata di gioco. In blu la prossima casella da raggiungere

L'oggetto mosso dall'utente (denominato "box") è ancorato alle intersezioni delle linee di una griglia quadrata. Gli unici momenti in cui si discosta da questi punti discreti è durante la fase di movimento, e viene riportato istantaneamente su quello più geometricamente vicino nel momento in cui l'utente termina l'interazione (solleva il dito dal tasto di trascinamento, smette di toccare il touchscreen, etc.). Similmente, il punto bersaglio viene scelto soltanto tra le posizioni stabili raggiungibili dal box, e viene contato come raggiunto nel momento in cui il giocatore termina l'interazione avendo esso come punto più vicino, portando quindi il box sulla sua stessa posizione.

La griglia di posizioni possibili del bersaglio è un quadrato di lato dispari $2L+1$, che comporta un numero di movimenti relativi possibili di $(2L+3)^2-1$ (è escluso il movimento stazionario).

Si assume che, indipendentemente dal tipo di dispositivo di input usato dall'utente, questo generi una serie di dati di movimento caratterizzati dalla tripletta *Istante di tempo*, *Delta X*, *Delta Y*. Chiamiamo queste triplette **micromovimenti**.

Formalmente, ogni ripetizione del loop di interazione segue questo flusso:

1. Viene estratta una nuova posizione per il bersaglio (diversa da quella del box)
2. Il giocatore inizia l'interazione (preme il tasto di trascinamento, tocca lo schermo, ...)
3. Il giocatore manovra il box verso il bersaglio usando il dispositivo di input

4. Il giocatore termina l'interazione, lasciando che il box venga riposizionato sul punto discreto più vicino
5. Se il punto di cui sopra non è quello bersaglio si torna al punto 2. Altrimenti, al punto 1 Il gioco può essere concluso in qualsiasi momento e dopo un numero qualsiasi di ripetizioni.

La seconda funzionalità dell'applicazione è il salvataggio dei dati generati dalle interazioni dell'utente. Si assume che questi vengano generati in un'unica sessione di gioco in cui l'utente ha effettuato movimenti il più efficienti possibili dalla partenza al bersaglio (senza deviazioni superflue) e che contengano i seguenti valori (o informazione sufficiente per calcolarli):

- Il numero N di sequenze totali
- Il numero di micromovimenti M_j della sequenza (con $j \in 1..N$)
- L'istante t_{jk} dall'inizio interazione e la direzione v_{jk} del micromovimento k della sequenza j
- Il movimento totale (in coordinate della griglia) w_j della sequenza j

Un dataset adeguato dovrebbe contenere esempi di sequenze per ogni possibile movimento relativo totale, quantità che scala con il quadrato del lato dello spazio di possibili posizioni del bersaglio. L'ipotesi fondamentale di questa implementazione della comunicazione semantica è che sarà possibile associare la sequenza di triplette al movimento totale sulla griglia usando tecniche di machine learning.

1.7: Applicazione per raccolta dati – Implementazione

L'implementazione dell'applicazione è stata eseguita nel motore di gioco Godot versione 4.0 (godotengine.org) usando il linguaggio C#. L'interazione utente è possibile usando un mouse o un eventuale trackpad integrato al dispositivo. In via preliminare, L è fissato a 1 per minimizzare il tempo di raccolta dati.

Il logging dei dati avviene in formato `.toml` utilizzando la funzionalità integrata del motore per salvare dati di gioco personali. Una sequenza memorizzata è riportata in Fig. 4.

```
[[movements]]                # elemento della lista "movements"
motion_duration = 283        # durata della sequenza in ms
max_displacement = 1         # => 9 posizioni possibili per il bersaglio
start_pos = [-1,1]
end_pos = [1,0]
relative_movement = [2,-1]   # movimento relativo totale
target_reached = true        # se falso: sequenza ignorata
sequence_length = 17         # numero totale di triplette
```

```
sequence = [
    [0, 1, 0],
    [16, 5, -2],
    [33, 9, -3],
    [50, 21, -4],            # [timestamp (ms), delta_x, delta_y]
    [66, 33, -13],
    [83, 42, -15],
    [100, 38, -16],
    [116, 60, -24],
    [150, 24, -6],
    [166, 10, -1],
    [183, 2, 0],
    [200, 1, 0],
    [216, 2, -1],
    [233, 5, -2],
    [250, 4, -3],
    [266, 10, -6],
    [283, 3, -1],
]
```

Figura 4: Esempio di sequenza salvata in memoria

Capitolo 2: Modello K-means

Una volta ottenuto un numero sufficiente di dati nel file di output è possibile procedere alla loro analisi. Obiettivo di questa fase è usare tecniche di machine learning per classificare accuratamente le sequenze di triplette e assegnarle al corretto movimento relativo totale.

2.1: Il problema

La procedura di raccolta dati ha prodotto una sequenza di tensori interi bidimensionali di forma non costante $(3, M_j)$, ognuno corrispondente a un movimento complessivo univoco. Affinché il dispositivo speculativo possa riconoscere il gesto dell'utente, dobbiamo verificare che esiste un processo di machine learning che è in grado, utilizzando una quantità sufficiente di dati di addestramento, di associare in maniera affidabile le sequenze ai movimenti complessivi.

A seconda che siano disponibili esempi annotati con la risposta desiderata o meno, gli attuali algoritmi di machine learning si dividono rispettivamente in apprendimento *supervisionato* o *non supervisionato*. É nell'interesse del progetto dimostrare che è possibile ottenere previsioni corrette affidandosi unicamente ai secondi, poiché questo semplificherebbe la comunicazione tra dispositivo e macchina (al dispositivo di input basterebbe raccogliere dati di esempio, classificarli e inviare il movimento complessivo al sistema operativo senza comunicazioni in senso opposto).

Le implementazioni più comuni degli algoritmi di questo tipo (con l'eccezione di eventuali parametri aggiuntivi) hanno la stessa interfaccia: nella visione "a scatola nera" viene dichiarato un oggetto (a cui sono associati, se presenti, i parametri specifici dell'algoritmo) che rappresenta il modello. Questo viene addestrato tramite una chiamata a funzione `.fit(X)` che ha come parametro una collezione di dati (descritta come una matrice bidimensionale di forma `(n_samples,n_features)`), che partiziona i dati di ingresso in *cluster* numerati. La quantità di cluster può essere decisa durante l'inizializzazione o conseguire dai parametri di ingresso. Una volta effettuato l'addestramento, l'oggetto può essere interrogato su dati appartenenti allo stesso spazio vettoriale restituendo l'indice del cluster corrispondente tramite una funzione `.predict(X)`, dove X è una lista di punti e l'output è la lista degli indici corrispondenti.

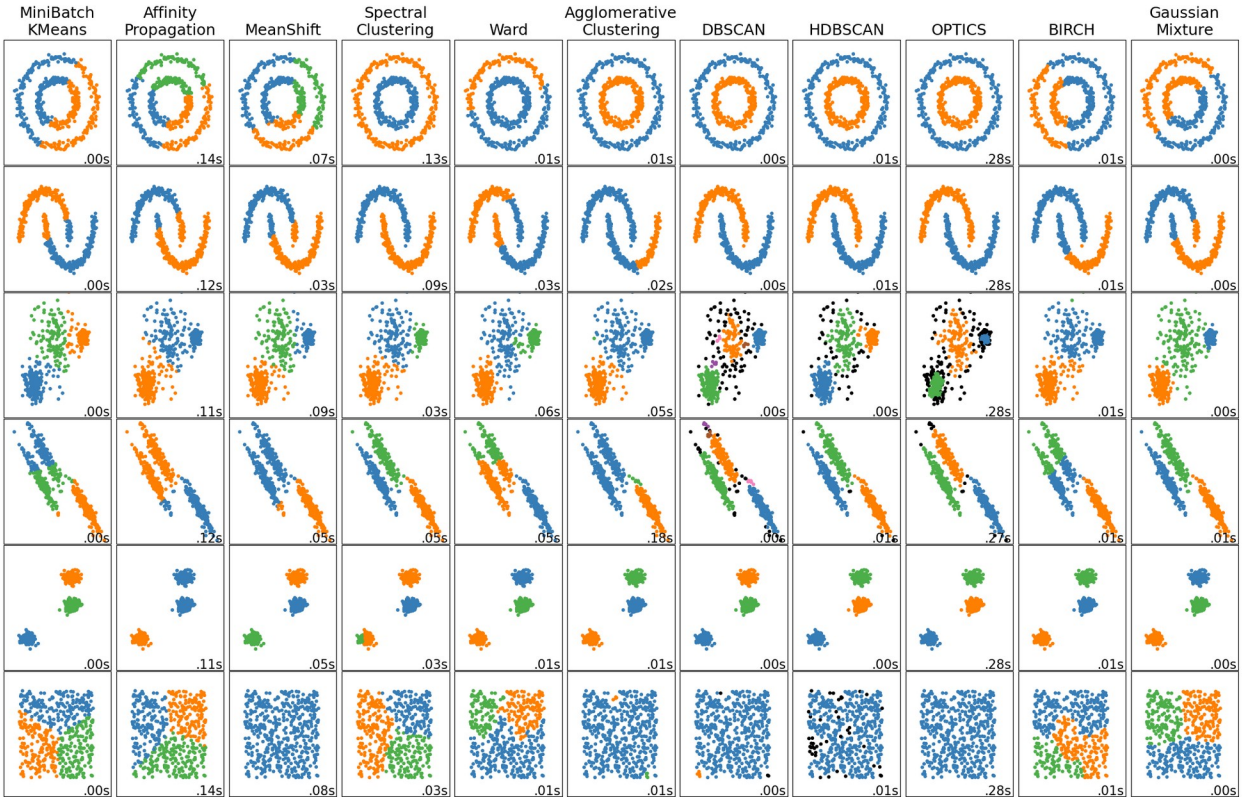

Figura 5: Confronto tra algoritmi di clustering della libreria scikit-learn

Ci aspettiamo che i dati abbiano una correlazione spaziale all'interno della classe e siano distribuiti in "nuvole" attorno a un punto medio.

2.2: Adattamento dati

Per conformare i dati a quelli richiesti in input dagli algoritmi sono richieste alcune trasformazioni delle sequenze di micromovimenti. Le seguenti operazioni sono espresse in forma concatenata, ognuna avente in ingresso l'uscita della precedente.

Forma della matrice: I dati iniziali sono una lista di N sequenze di forma $(3, M_j)$.

```
[
    [t1, x1, y1],
    [t2, x2, y2],
    ... ,
    [tM, xM, yM]
]
```

Per renderli monodimensionali è sufficiente accodare le triplette

```
[
      t1, x1, y1, t2, x2, y2, ... , tM, xM, yM
]
```

Lunghezza delle sequenze: Le sequenze attuali hanno lunghezza eterogenea $3M_j$. Per poter addestrare il modello è necessario che i punti appartengano allo stesso spazio vettoriale, il che è

ottenuto applicando un *padding* di zeri che porta la lunghezza a una costante uguale o superiore a quella della sequenza più lunga ragionevolmente ottenibile. Assumiamo che M_{max} sia il numero massimo di triplette consentito.

```
[t1, x1, y1, ... , tM, xM, yM] => [t1, x1, y1, ... , tM, xM, yM, 0, 0, 0, ...]
```

Le N sequenze risultanti hanno forma `(3 * M_max,)` e vengono chiamate X_j o, intercambiabilmente, righe della matrice X (di forma `(N, 3*M_max)`) o *punti dello spazio vettoriale*.

2.3: Il modello K-means

Tra le varie scelte possibili di algoritmi per il clustering preliminare è stato scelto K-means, utile per suddividere punti appartenenti a un numero conosciuto di classi distribuite in modo convesso senza sovrapposizioni in uno spazio. Esso si basa su un procedimento iterativo di spostamento di *centroidi* (punti nello stesso spazio vettoriale dei dati, corrispondenti al baricentro/media delle varie classi) e di partizione dei punti volta a minimizzare una certa funzione (solitamente lo scarto quadratico medio totale tra i punti e i centroidi corrispondenti).

Sia C l'insieme dei centroidi. Ogni centroide $c \in C$ corrisponde univocamente a una classe di movimento relativo.

K-means è un *algoritmo parametrico di apprendimento non supervisionato*. I parametri interni che variano durante l'addestramento corrispondono alle coordinate dei centroidi (e sono quindi $3M_j$ per ogni centroide).

La funzione *loss* (di perdita) da minimizzare ad ogni iterazione è comunemente implementata usando l'*inerzia*, ovvero la somma totale dei quadrati delle distanze intra-classe dei punti dai centroidi[2]. Presa una certa matrice X e una configurazione di centroidi C :

$$L(X, C) = \sum_{j=0}^{N} min_{c \in C}(\|X_j - c\|^2)$$

L'implementazione più comune (e quella utilizzata nella libreria scikit-learn) di K-means è tramite l'algoritmo di Lloyd, come descritto in fig. 6.

```
# INPUT:
#   x1, x2 := punti nello spazio vettoriale a n dimensioni
#             (valori accessibili con x1[i])
# OUTPUT:
#   d := distanza quadratica tra i due punti (numero reale positivo)
```

2 https://scikit-learn.org/stable/modules/clustering.html#k-means

```
FUNCTION distance (x1, x2)

    d <- 0
    foreach (i in 0..n)
        d <- d + (x1[i]-x2[i])^2
    endfor
    return d

END FUNCTION

---

# INPUT:
#   K := numero di cluster in cui partizionare lo spazio
#   X := matrice dei dati di addestramento di forma (N, 3*M_max)
#   C_init := (opzionale) posizione iniziale dei centroidi (matrice di forma
(K,3*M_max))
# OUTPUT:
#   C := matrice di forma (K,3*M_max) di centroidi

FUNCTION KMeans (K, X, C_init<-null)

    C <- C_init

    if (C = null)
        C <- (matrice di valori scelti casualmente o secondo un'euristica)
    endif

    new_C <- (matrice con stessa forma di C ma valori diversi per entrare nel
loop)

    while (new_C != C)  # le iterazioni terminano quando si è raggiunto l'ottimo
locale (o una soglia sufficiente)

        C <- new_C

        clusters <- (lista di K liste vuote di punti)
        foreach (x in X)

            mindist <- +inf
            min_centr <- 0

            foreach (i in 0..length(C))
                dist <- distance(x,C[0])
                if (dist < mindist)
                    mindist <- dist
                    min_centr <- i
                endif
            endfor

            clusters[i].append(x)

        endfor

        foreach (i in 0..K)
            new_C[i] <- (media geometrica dei punti di cluster[i])
        endfor
```

```
    endwhile

    C <- new_C
    return C

END FUNCTION
```

Figura 6: Pseudocodice dell'algoritmo e della funzione distanza usata al suo interno

L'algoritmo suddivide in ogni iterazione la lista di punti sulla base del centroide più vicino (stabilito tramite la funzione distanza, in questo caso la norma quadrata). Una volta trovata la lista di punti per ogni classe, esso calcola i nuovi centroidi effettuando la media dei punti. Le iterazioni continuano fino al raggiungimento della stabilità, ovvero quando la posizione dei centroidi non cambia più a seguito dell'iterazione.

Il risultato dell'applicazione di questo algoritmo è un raggiungimento di un minimo locale della funzione Loss sopra descritta. Come molte altre funzioni di ottimizzazione, K-means è soggetta a possibili errori di mancato raggiungimento del massimo globale, quindi molte implementazioni (tra cui quella di `sklearn` in uso in questa analisi) applicano alcune misure per mitigare le possibilità di errore, tra cui l'inizializzazione di centroidi distanti tra loro (`kmeans++`) e la riapplicazione con centroidi iniziali differenti (controllato dal parametro `n_init`, default 10).

2.4: Interfaccia del modello

Sia F un modello addestrato come descritto sopra e X_j un vettore movimento corrispondente alla classe di spostamento $c \in C$. L'interrogazione del modello usando come input il vettore, scritto come $F(X_j)$, restituisce una classe $c' \in C'$, dove C' è una rimappatura di C con stessa cardinalità e corrispondenza biunivoca tra classi dei due insiemi. In uno scenario reale, la conversione di classi tra i due insiemi è lasciata all'applicazione.

2.5: Valutazione del modello

Obiettivo di questa fase è determinare quanto questo formato di dati si presta a un clustering sferico e quali misure possono essere messe in atto per ottimizzare la classificazione. L'analisi è stata eseguita utilizzando il linguaggio Python 3.11, popolare nel campo di analisi dati, con l'ausilio delle librerie `numpy, scikit-learn.`

Il compito dell'algoritmo K-means è suddividere i punti forniti in cluster che speriamo essere quanto più simili possibili ai gruppi originali di spostamenti. Tuttavia, i cluster di assegnamento appartengono a classi astratte per cui non viene fornita un'associazione intrinseca agli spostamenti

reali (associazione che, per gli scopi di utilizzo di questo sistema, non sarebbe nemmeno necessaria). Tuttavia, ai fini della misurazione delle prestazioni dell'algoritmo, ci avvaleremo delle etichette dello spostamento reale assegnate automaticamente dal gioco. Questo ci consente di computare il movimento medio $\bar{X}_c$ per ogni classe, che assumiamo essere rappresentativo degli altri elementi. Chiamiamo C_{map} l'insieme delle classi ottenuti dalla mappatura $F(\bar{X}_c) \forall c \in C$.

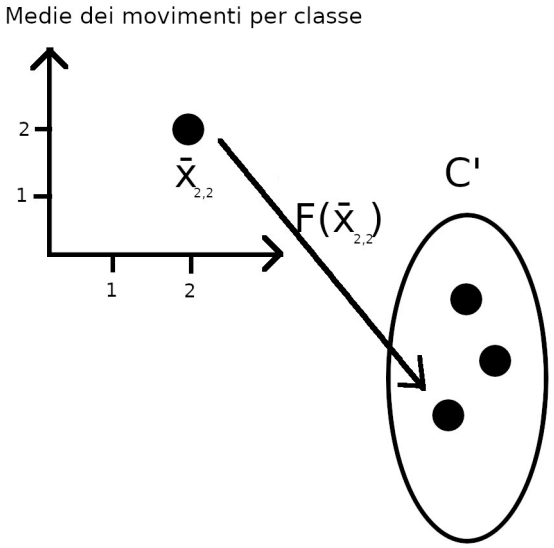

Figura 7: Assegnamento di uno spostamento a un cluster

Ci avvaliamo di due metriche per valutare l'efficacia del modello. La prima, che chiameremo *cluster loss*, rappresenta la differenza tra la cardinalità degli elementi unici di C e di C_{map} . La seconda, valida solo in un modello che ha ottenuto cluster loss 0, sarà l'accuratezza di predizione della classe di una batteria di dati di test acquisiti separatamente da quelli di addestramento. Nella configurazione descritta sopra, la cluster loss massima è 23 (scenario in cui ogni classe converge su un unico centroide).

La classe restituita da un'interrogazione del modello è un indice intero senza legame esplicito allo spostamento a cui corrisponde. Ai fini della valutazione dell'accuratezza, avremo bisogno di assegnare temporaneamente ciascuno a uno spostamento concreto, cosa che possiamo ottenere interrogando il modello su ogni media di classe $\bar{X}_c$. Questo ci consente di assegnare una classe prevista c' per ogni punto dello spazio vettoriale appartenente alla sequenza di test e valutarne la correttezza.

2.6: Addestramento del modello

Allo scopo dell'addestramento sono stati rilevati circa 2700 serie di movimenti, con almeno 100 rappresentanti ogni tipo di classe.

Il modello addestrato come descritto sopra si è rivelato sorprendentemente poco performante, con una cluster loss media di 15. Sono stati necessari diversi accorgimenti per abbassare quel valore a 0, descritti a seguire.

Elisione del dato temporale: Un'analisi attenta dei dati raccolti ha rivelato che il campionamento avveniva con periodo di circa 15 millisecondi, dando ai vettori un aspetto molto ridondante:

```
[ 0-1 x1 y1 15-16 x2 y2 30-32 x3 y3 ...]
```

Questo avvicinava di molto i dati su un terzo degli assi, confondendoli ai fini del clustering. Eliminare i dati temporali dalla sequenza (riducendo la forma della matrice X a `(N,2*M_max)` ha immediatamente abbassato la cluster loss media a 7.

Distanziamento caselle: Il motore di gioco Godot misura la distanza tra oggetti sul piano 2d in pixel. Alzare il distanziamento tra centri delle caselle da 128 a 200 ha abbassato ulteriormente la cluster loss media a 4.

Iterazione: Il modello è stato lasciato in addestramento per un periodo prolungato fino all'ottenimento di una cluster loss nulla. Per evitare la necessità di ripetere l'intero processo, la posizione dei centroidi è stata salvata in un file di testo.

Una volta ottenuto un insieme C_{map} con cluster loss 0 possiamo passare alla valutazione della performance di predizione. Queste sono esposte nella sezione 4.1.

Capitolo 3: Il modello neurale

Viene descritto ora in dettaglio il modello introdotto nella sezione 1.3. In modo simile al primo modello, il suo compito sarà prendere in input il vettore movimento e restituire una previsione dello spostamento corrispondente, con la differenza sostanziale che l'input dovrà essere il sottoinsieme iniziale di lunghezza minima del vettore movimento. Chiameremo questo sottoinsieme una *troncatura*.

3.1: Notazione

Dato il vettore X_j , definiamo la troncatura $T(X_j, n)$ come il vettore ottenuto prendendo le prime n coppie di X_j , ovvero i suoi primi 2n elementi, mantenendone l'ordine. Dato che la troncatura coinvolge soltanto i dati in ingresso, le corrispondenti classi $c \in C$, comunemente chiamate etichette o *label*, non dovranno subire modifiche se non nella codifica richiesta dall'architettura del modello.

3.2: Descrizione modello

Sia G_n il modello addestrato a partire da questi dati. L'interrogazione $G_n(T(X_j, n))$ restituisce un vettore $\hat{C}_j$, di lunghezza pari al numero di classi, che rappresenta la confidenza del modello in ogni possibile risposta (ad esempio, per $|C| = 2$, un vettore $\hat{C} = [0.8, 0.1]$ rappresenta una confidenza dell'80% nella prima risposta e del 10% nella seconda). Una risposta singola $\hat{c}_j$ è ottenuta applicando $argmax(\hat{C}_j)$.

Per consentire di comparare le etichette al risultato finale, queste saranno convertite nel formato *one-hot*, dove un elemento appartenente a un insieme ordinato finito è espresso con una sequenza di valori binari (0 o 1) dove l'unico valore non nullo corrisponde all'ordine dell'elemento.

Il modello è costruito utilizzando una rete neurale costituita da una sequenza di strati, o *layer*, densi. Lo strato di input accetta vettori di lunghezza 2n, quello di output restituisce vettori di lunghezza $|C|$.

Tra questi due strati sono frapposti un certo numero di strati intermedi che hanno il compito di apprendere certe feature della sequenza e, a partire da queste, il significato della troncatura da esprimere in termini di confidenza della risposta. La funzione di attivazione di questi strati, in quanto espressioni di feature non negative, sarà l'unità lineare rettificata, mentre quella del tensore in output, dovendo essere un intervallo di confidenza percentuale, sarà in alternativa la funzione sigmoide o la funzione *softmax*.

Gli algoritmi di apprendimento delle reti neurali convenzionali richiedono inoltre una funzione di *loss* che consenta di stimare la distanza della risposta del modello dall'etichetta (e di conseguenza rendere possibile la retropropagazione dei pesi). Per i formati di output one-hot questa è solitamente l'*entropia incrociata binaria*, ma tipi più specifici di classi di spostamento che godono di proprietà addizionali possono implementare funzioni personalizzate.

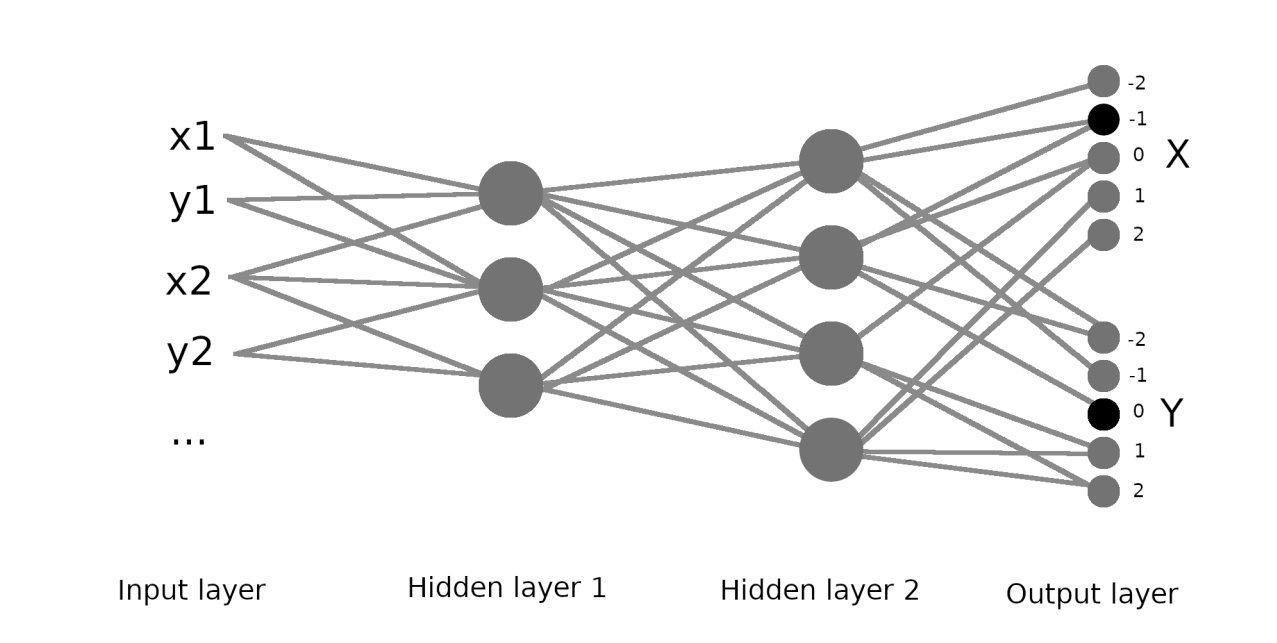

Figura 8: Struttura della rete neurale costruita
(Nell'immagine, una configurazione con due strati intermedi da 3 e 4 nodi)

3.3: Implementazione modello

Il modello è costruito utilizzando la libreria `tensorflow` di python, opzione estremamente popolare nel campo del machine learning. In particolare, viene fatto uso del sottomodulo `keras`, utile per una definizione concisa e dichiarativa della rete.

Dato che il gioco implementato per questo progetto ha classi di spostamento esprimibili con una coppia di valori (x,y), entrambi interi compresi tra -2 e 2, abbiamo scelto di separare le due componenti nell'output del modello, con una funzione loss basata sulla distanza di manhattan valutata separatamente per ciascuna componente. Una rappresentazione stilizzata del modello è illustrata in figura 8.

La funzione loss in uso è definita in fig. 9. La perdita finale è valutata in base alla somma della distanza di ogni risposta dall'etichetta vera moltiplicata per la confidenza del modello in essa, a cui si somma la mancanza di confidenza del modello nella risposta giusta.

```
# INPUT:
#    label_one_hot := etichetta (es. [0 1 0 0 0])
# OUTPUT:
#    label_field_distance := array di distanza di campo (es. [1 0 1 2 3])

FUNCTION field_distance(label_one_hot)

    label_field_distance <- (array di zeri con stessa lunghezza dell'input)
    index <- argmax(label_one_hot)
    for i in 0..length(label_one_hot):
        label_field_distance[i] <- abs(i-index)
    endfor
    return label_field_distance

END FUNCTION

---

# INPUT:
#    label_true := risposta attesa dal modello
#    label_pred := risposta restituita dal modello
# OUTPUT:
#    loss := misura numerica della distanza dalla risposta corretta

FUNCTION manhattan_loss(label_true,label_pred)

    true_field_dist <- field_distance(label_true)
    loss <- label_pred * true_field_dist # prodotto element-wise
    loss <- loss + label_true * (1 – label_pred)

    return sum(loss)

END FUNCTION
```

Figura 9: Pseudocodice della funzione loss usata per il modello

3.4: Addestramento

Come dati in ingresso sono stati impiegati gli stessi usati per il modello K-means del capitolo 2. L'addestramento è stato ripetuto con varie configurazioni di strati e per un numero n di coppie da 1 a 20.

La libreria in uso consente inoltre di applicare diverse funzioni *callback* al termine di ogni iterazione di addestramento, includendo implementazioni di default per quelle più usate. Di particolare interesse è stata la funzione `keras.callbacks.ReduceLROnPlateau`, che abbassa temporaneamente il tasso di apprendimento del modello dopo un certo numero di iterazioni consecutive in cui non ci sono stati miglioramenti.

3.5: Valutazione

Durante la compilazione del modello è inoltre possibile definire di quali metriche tenere conto per la valutazione della qualità complessiva. La più importante ai nostri scopi (e inclusa come opzione di default per i modelli) è l'*accuratezza categorica*, la percentuale di casi in cui il risultato più confidente del modello coincide con il valore non nullo dell'etichetta. Questa sarà valutata separatamente per le componenti x e y.

La misurazione verrà effettuata al variare della lunghezza n di troncatura.

Capitolo 4: Risultati

4.1: Modello K-means

Come discusso in 2.3, la valutazione del modello (solitamente agnostico sull'ordinamento della classe di spostamenti che restituisce) ha richiesto di assegnare temporaneamente un'etichetta concreta alla classe numerica restituita. Questo è stato ottenuto computando il movimento medio per-classe e notando il centroide assegnato dal modello.

La valutazione è stata poi effettuata rilevando una seconda serie di circa 1000 dati e valutando la percentuali di questi che il modello assegnava allo stesso centroide della media della loro classe. Questa percentuale si è rivelata essere l'85%, un risultato non ottimale ma che dimostra la possibilità di ottenere risultati migliori con modelli più sofisticati.

È possibile inoltre effettuare una valutazione qualitativa della bontà del modello: si può pensare a un addestramento di un modello non supervisionato come a una scatola nera che, dati in input una serie di dati, li suddivide in gruppi di elementi simili tra loro. Dato che K-means valuta gli elementi per prossimità spaziale, possiamo aspettarci che i centroidi prodotti siano rappresentativi del movimento corrispondente. Fortunatamente per gli scopi di questa analisi, i centroidi non sono che un'istanza del tipo di dati di partenza, ovvero una serie di movimenti relativi bidimensionali. Questo rende particolarmente facile sommare iterativamente questi micromovimenti per ottenere il percorso complessivo, come mostrato nella fig. 10 (ottenuta grazie alla libreria `matplotlib`). Nella figura possiamo osservare come, a scanso di idiosincrasie in particolari movimenti, ogni centroide codifica un movimento quasi rettilineo dalla casella di partenza a quella di destinazione ed è possibile distinguere visualmente quale movimento corrisponde a quale spostamento.

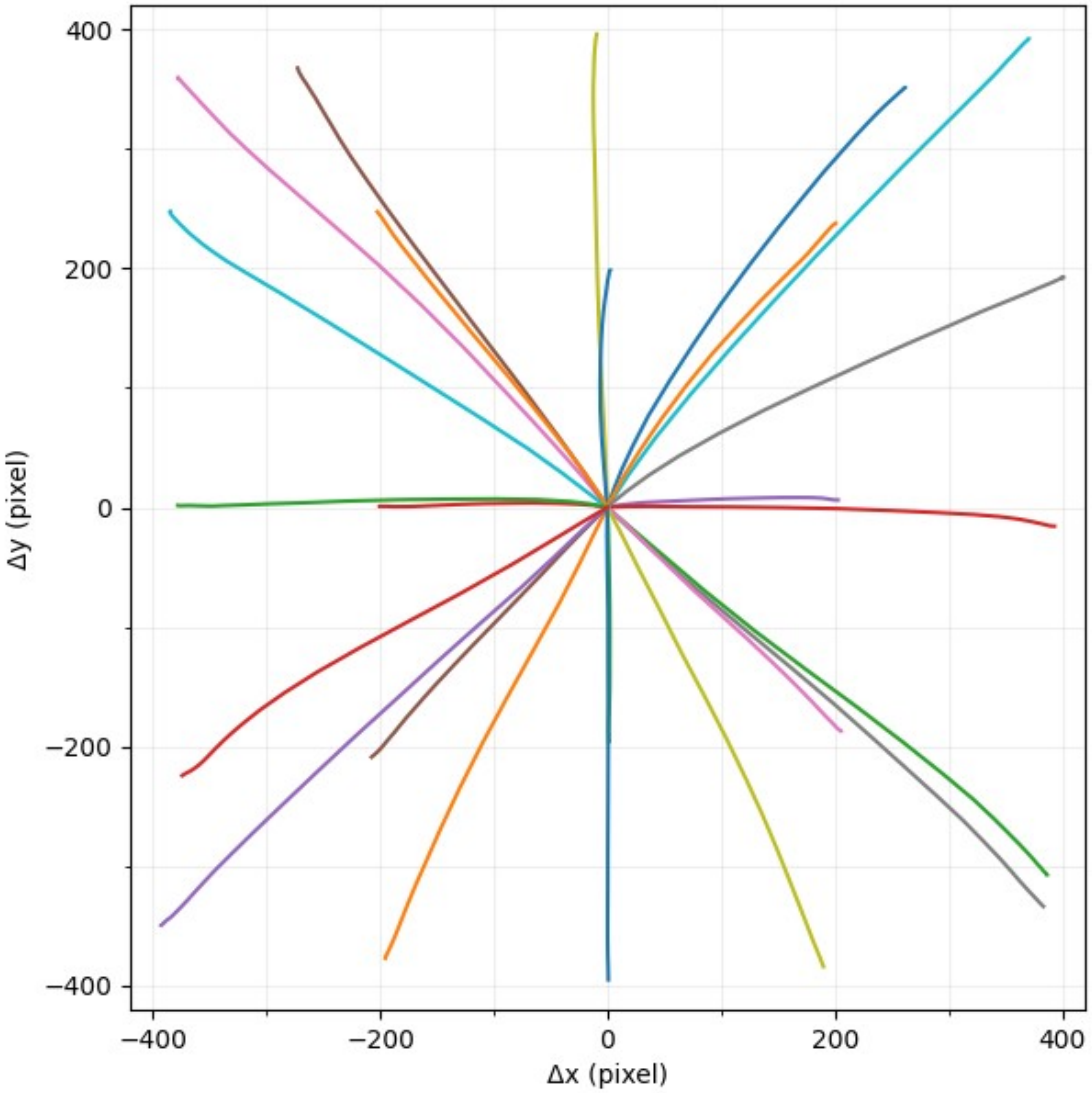

Figura 10: Visualizzazione dei movimenti appresi dai cluster

4.2: Modello neurale

La performance della rete neurale si è rivelata deludente, pur essendo migliore di una stima puramente casuale. Nonostante l'impiego di tipi di architettura a vari livelli di profondità e numero di nodi, l'accuratezza delle previsioni per ogni componente non ha mai superato con consistenza il 70%. Al variare della lunghezza di troncatura, inoltre, si sono osservate fluttuazioni imprevedibili del valore tra il 30% e il 70%, e la variazione della performance delle due componenti appare non correlata.

Una possibile ipotesi dell'incapacità della rete di eccedere questo livello di prestazioni è l'impossibilità di questa tipologia di modelli di gestire dati caratterizzati da una componente temporale. Questo impedirebbe al modello di estrarre informazioni sufficienti dalla sequenza.

Nelle figure 11 e 12 sono mostrate le variazioni di performance per componente e medie in base alla lunghezza di troncatura. La struttura di ogni rete è espressa nel titolo del grafico (per esempio, una rete 32-6 possiede due strati intermedi dotati rispettivamente di 32 e 6 nodi).

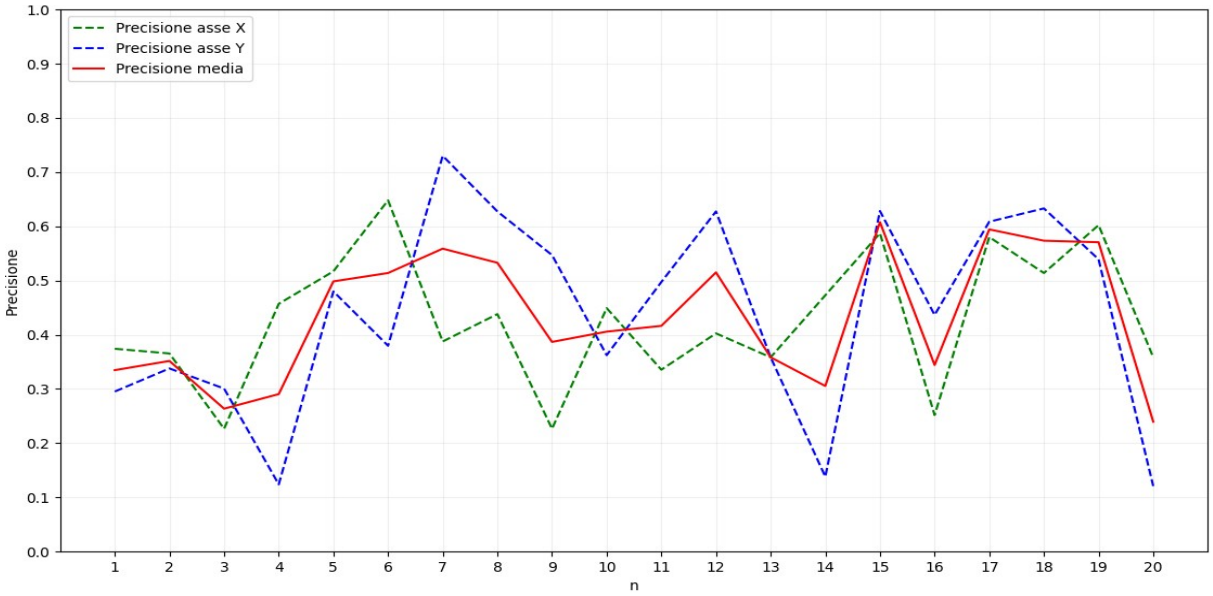

Figura 11: Performance della rete 32-32-6

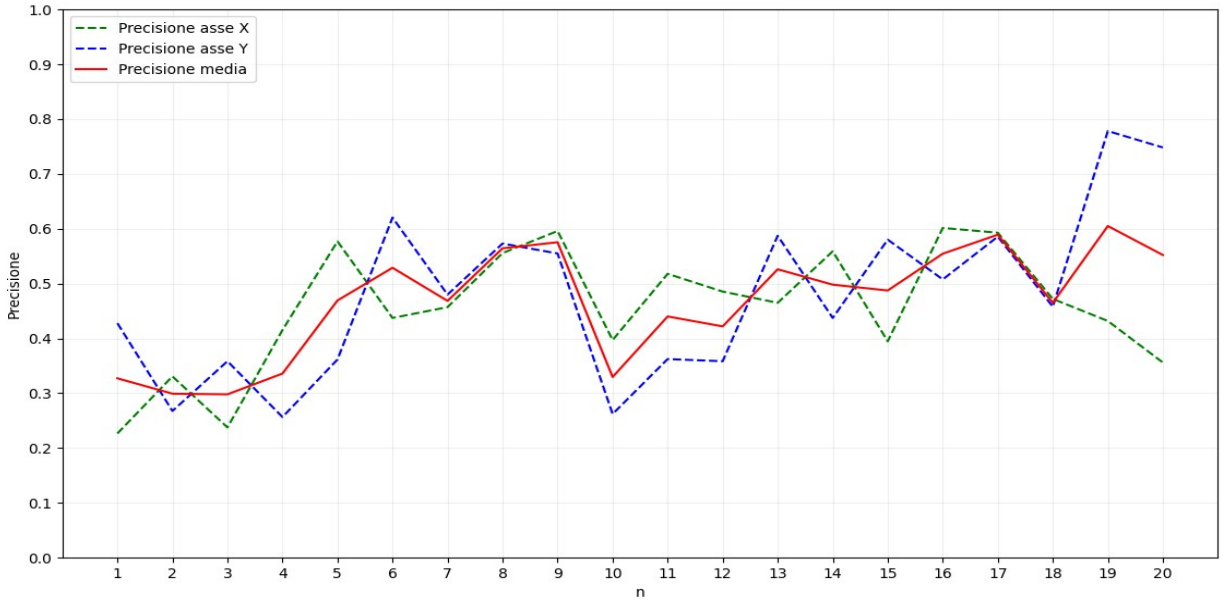

Figura 12: Performance della rete 32-10-6

Capitolo 5: Conclusioni

5.1: Considerazioni sui risultati ottenuti

Il modello costruito secondo l'algoritmo K-means, pur essendosi rivelato più performante del modello neurale, è stato inteso unicamente come un metodo di raggruppamento ai fini dell'addestramento di modelli più avanzati. La necessità di prendere in input il movimento completo significa che questo non offre previsioni anticipate dello spostamento totale, riducendosi a una semplice compressione del messaggio in uscita dal dispositivo.

Pur non avendo prodotto un modello capace di estrarre il significato della sequenze di movimenti in modo affidabile, questo progetto ha rivelato dati incoraggianti sulla possibilità teorica di esso. Fattore limitante principale è stata la mancanza di esperienza nella costruzione di reti neurali, essendo questa disciplina basata sulla decisione di architettura e taratura di parametri con conseguenze ramificate e non facilmente intuibili.

5.2: Possibilità di indagini future

La portata del nostro lavoro si è limitata all'utilizzo di sole due tecniche di apprendimento automatico (K-means e reti neurali dense), ma i risultati ottenuti suggeriscono che altri tipi di modelli potrebbero trovare più successo nei compiti di raggruppamento e previsione degli spostamenti.

Dal punto di vista del raggruppamento, i modelli a Macchine a Vettori di Supporto (SVM), discriminando per bisezioni successive dello spazio vettoriale invece di prossimità spaziale, potrebbero ottenere risultati più accurati al costo di un maggior numero di parametri da impostare prima dell'addestramento.

Per quanto riguarda la previsione da troncatura, i risultati ottenuti precedentemente non escludono che possa esistere un'architettura di rete neurale che abbia prestazioni uguali o superiori al livello desiderato. Per esempio, sostituendo un modello statico con uno dinamico, un modello a Rete Neurale Ricorsiva (RNN) dotato di Long Short Term Memory (LSTM), potrebbe avere più successo nella previsione di spostamenti totali. Questo tipo di modello è dotato di nodi che memorizzano il loro stato e, una volta istanziato, viene simulato in una sequenza di istanti temporali che possono accettare come input l'output dell'istante precedente. Un modello di questo tipo potrebbe ricevere un micromovimento in input per ogni istante e fornire una previsione gradualmente sempre più accurata dello spostamento.

Bibliografia

- K. Niu et al., “A Paradigm Shift toward Semantic Communications”, IEEE Commun. Mag., Nov. 2022
- Z. Qin et al., “Semantic Communications: Principles and Challenges”, arXiv:2201.01389
- Harris, C.R., Millman, K.J., van der Walt, S.J. et al. “Array programming with NumPy”. Nature 585, 357–362 (2020). DOI: 10.1038/s41586-020-2649-2
- J. D. Hunter, "Matplotlib: A 2D Graphics Environment", Computing in Science & Engineering, vol. 9, no. 3, pp. 90-95, 2007
- Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011
- TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org

Ringraziamenti

Ringrazio innanzitutto il professor Tomasin, che ha tenuto questo progetto in carreggiata.

Ringrazio di cuore i miei genitori, il cui supporto è stato fondamentale, le mie sorelle, che nonostante gli occasionali conflitti sono la squadra migliore che si possa desiderare, i miei nonni, che mi hanno donato più di quanto potrò mai restituirgli, tutti i miei familiari e la mia gatta Coco.

Ringrazio i miei amici, sia vecchi che nuovi, che mi hanno dato sostegno incondizionato e che hanno sopportato quanto questo lavoro mi abbia occupato negli ultimi mesi. Tra loro ringrazio in particolare quelli con cui spero un giorno di guadagnare da vivere sviluppando videogiochi, e quelli che mi hanno aiutato sul tecnico con le parti meno intuitive del machine learning.

Ringrazio il Laboratorio Sociale Occupato “La Tana” per l’incredibile comunità che mi ha consentito di trovare, e ringrazio in particolare il gruppo di Critical Psychology, con cui ho vissuto momenti indimenticabili.

Ringrazio il coro giovanile SingOverSound di Vittorio Veneto, a cui spero di ritornare il prima possibile.

Ringrazio il concetto astratto di prog rock e il peluche a forma di squalo preso all’IKEA.

Voglio anche ringraziare tre videogiochi e i loro rispettivi autori:

- TIS-100, per avermi fatto realizzare che volevo studiare ingegneria informatica
- Outer Wilds, per il suo amore incondizionato per la curiosità
- Disco Elysium, per ricordarmi che un mondo migliore è possibile