# POLITECNICO DI TORINO

Corso di Laurea Magistrale in
Ingegneria Informatica

Tesi di Laurea Magistrale

# Analisi delle metriche di similarità per il riconoscimento di firme false incrociando dataset, lingua e alfabeto

**Relatore**
prof. Tatiana Tommasi

**Candidato**
Luca Vezzani

**Tutor aziendale**
Mirko Zaffaroni

Dicembre 2020

# Dedica

Un sentito ringraziamento va alla professoressa Tatiana Tommasi, per la sua competenza e disponibilità, e a Mirko Zaffaroni per il prezioso e continuo supporto.

Grazie a tutte le persone che, in qualche modo, mi hanno accompagnato in questa esperienza, faticosa ma splendida: grazie ai compagni di corso, con i quali ho condiviso l'ansia e la gioia dei singoli traguardi; grazie agli amici, quelli vicini che hanno compreso le mie assenze dando invece valore alle volte in cui c'ero, e a quelli lontani ma di sempre, presenti nei momenti importanti, la cui amicizia è invulnerabile al passare del tempo.

Grazie a Elena, compagna che è stata sempre presente, ultima a perdere la speranza nei momenti difficili e prima a spronarmi ad andare avanti: senza di lei, questo percorso sarebbe stato molto più ostico.

Infine, vorrei ringraziare i miei genitori: mi hanno visto intraprendere una sfida più grande di me, sapendo quanto sarebbe stata impegnativa; nonostante questo, hanno deciso di accompagnarmi passo dopo passo, condividendo in silenzio le paure e le preoccupazioni, senza però mai dubitare di quello che sarebbe stato il risultato finale.

Grazie.

# Sommario

La verifica della firma offline è uno dei task più complessi nell'ambito della biometrica e per quanto riguarda la ricerca forense dei documenti. La verifica è resa difficile da due fattori principali: l'originale non è uno standard sempre identico, ma è dipendente da fattori che possono influenzare il firmatario come stato psicologico, la penna che sta usando, la posizione, e altre condizioni dettate dall'ambiente; il falso a volte corrisponde all'originale a meno di dettagli difficilmente percettibili. In questa tesi si propone un modello di verifica della firma offline basato su rete neurale artificiale siamese con particolare attenzione alla funzione utilizzata per determinare l'apprendimento. La rete siamese è una struttura con due reti gemelle che condividono i pesi delle sinapsi che collegano i vari neuroni, unità di elaborazione della rete stessa. L'attenzione focalizzata sulla funzione di costo nasce dall'analisi dell'importanza della "distanza" che si cerca di stabilire tra coppia di firme originali e coppia di firme dove una delle due risulta essere forgiata, distanza la cui enfatizzazione dipende molto dal tipo di dato a cui viene applicata. Gli esperimenti di questo elaborato si concentrano sui risultati ottenuti quando, ad un modello ottimizzato su una lingua, vengono sottoposte firme in altre lingue da verificare, task che ha riscontrato risultati inferiori se confrontati con la controparte rappresentata dalla verifica di firme appartenenti alla stessa lingua. Il modello proposto in questo elaborato ha superato i risultati riportati su altri paper presenti nello stato dell'arte attuale nell'ambito del riconoscimento di firme forgiate incrociando lingua e alfabeti diversi.

# Indice

# Capitolo 1

# 1 Introduzione

La firma ha svolto un'importante ruolo nelle comunicazioni umane sin dagli albori della civiltà. Essa veniva utilizzata come metodo per identificare la proprietà di una persona, per distinguere prodotti simili fornendo informazioni su chi li aveva fabbricati oppure come simbolo su lettere o missive per confermare che il contenuto era legato ad uno specifico mittente (possiamo considerare i sigilli come precursori della firma moderna). Con l'evoluzione e l'espansione della società civile, la firma è stata ormai riconosciuta come metodo ufficiale utilizzato da un individuo per indicare la presa visione o per esprimere il suo consenso sul contenuto di un documento che gli viene presentato.

## 1.1 La falsificazione

Una delle principali problematiche che presenta questo strumento di riconoscimento così versatile è la falsificazione: non si parla solo di tentativi mediocri di riproduzione con l'obiettivo di ottenere i benefici che deriverebbero dal passare per il vero "proprietario" della firma, ma di professionisti che dedicano tempo e studi per far combaciare perfettamente il loro operato con l'originale. Per questo e per altri motivi, nel tempo si è resa necessaria l'istituzione di tecniche utilizzabili dalla scienza forense per poter tenere il passo dei cosiddetti "falsari": pause e interruzioni nella stesura, tratto iniziale e finale, esitazione nel tratto e pressione della penna sono diventati elementi chiave di studi scientifici [1].

## 1.2 ASV [Automatic Signature Verification]

Con l'avvento dell'automazione, la comunità scientifica si è subito mostrata interessata a proporre una soluzione al problema sopra indicato.
Tanto è stato l'interesse che, negli ultimi 40 anni, sono stati pubblicati dei veri e propri sondaggi sull'attuale stato dell'arte riguardante l'ASV, ovvero l'Automatic Signature Verification (Verifica Automatica della Firma). Fra molti paper pubblicati, alcuni di questi risultano aver avuto un impatto significativo:

- Uno dei primi lavori a rientrare in questa categoria è sicuramente quello pubblicato da Plamondon e Lorette nel 1989, "Automatic signature verification and written identification - the state of the art".; [2]

- Circa una decade dopo, lo stesso Plamondon insieme a Srihari pubblicarono un altro paper, "On-line and Off-line handwriting recognition: A comprehensive survey.", nel quale viene analizzata l'evoluzione delle tecnologie rispetto al paper precedente e viene introdotta un'analisi anche sul discorso della firma digitale; [2]

- Sempre mantenendo lo stesso spazio temporale, nel 2008 venne pubblicato un sondaggio completo da due ricercatori italiani, Impedovo e Pirlo, "Automatic Signature Verification, The state of the art."; [2]

- Ultimo paper a svolgere la funzione di sguardo completo sull'argomento è stato pubblicato grazie ad una collaborazione di sei ricercatori, alcuni autori dei precedenti, che nel 2019 hanno deciso di analizzare come l'evoluzione tecnologica sia nel campo dell'analisi che in quello dell'acquisizione digitale della firma avessero impattato lo stato dell'arte "A Perspective Analysis of Handwritten Signature Tecnology". [2]

## 1.3 Ruolo dell'intelligenza artificiale

Sicuramente di interesse per questo campo è stato l'avvento dell'intelligenza artificiale: questa ha introdotto strumenti molto efficienti per l'estrazione della firma dai documenti cartacei e la sua elaborazione in formato digitale.
Per quanto riguarda l'analisi di quest'ultima, il machine learning, branca corposa dell'intelligenza artificiale, ha mostrato un nuovo punto di vista sull'argomento: invece di imporre alla macchina come distinguere i falsi dall'originale, ha permesso ad essa di interpretare ciò che le veniva proposto e di identificare i tratti per lei importanti.
Nello specifico, il deep learning ha dimostrato di poter svolgere egregiamente questo compito, come illustrato da alcuni paper presenti nello stato dell'arte attuale:

- "SigNet: Convolutional Siamese Network for Writer Independent Offline Signature Verification", nel quale viene utilizzata una rete convoluzionale siamese per il confronto tra due firme;

- "Inverse Discriminative Networks for Handwritten Signature Verification", dove si propone un nuovo modello per permettere un focus maggiore sui tratti della firma.

In entrambi questi studi si può notare come la rete sia in grado di imparare molto bene a distinguere firme false quando l'apprendimento e il test riguardano firme appartenenti alla stessa lingua ma (e questo viene sottolineato dai risultati) incontri enorme difficoltà quando queste due fasi riguardano lingue diverse (ad esempio la rete venga allenata a riconoscere le firme false in inglese, ma il test venga eseguito utilizzando firme in Bengali).

## 1.4 Obiettivo e struttura di questa tesi

Lo scopo della mia tesi riguarda proprio questo aspetto: valutare l'efficacia di un modello di deep learning per il riconoscimento di firme falsificate nel caso in cui i dati di addestramento provengano da un dataset diverso rispetto ai dati di test.

Nei capitoli successivi descriverò brevemente il concetto di machine learning, cercando di non tralasciare gli aspetti chiave dell'argomento, per poter poi introdurre elementi più specifici quali il deep learning, le reti siamesi e il Metric learning, essenziali per comprendere gli esperimenti e i risultati mostrati in questo elaborato; dopodichè analizzerò più nel dettaglio lo stato dell'arte dal quale ho tratto spunto per questo lavoro, ed infine dedicherò due capitoli per descrivere la struttura implementata e commentare i risultati che ho ottenuto, paragonandoli anche ai lavori sopracitati.

# Capitolo 2

# 2 Machine learning

Si può affermare con tranquillità che negli ultimi anni, grazie ad un aumento della capacità di calcolo degli elaboratori e alla disponibilità sempre maggiore di dati, le tecniche di machine learning siano state l'oggetto di studio di molti ricercatori. Questo interesse ha permesso alla comunità scientifica di approfondire vari aspetti dell'argomento che si sono tradotti in applicazioni su vari campi, tra cui quello dell'analisi delle immagini, ambito portante degli esperimenti condotti in questa tesi. Come anticipato, in questo capitolo cercherò di spiegare cosa è il machine learning fornendo una visione d'insieme che possa facilitare la comprensione del resto dell'elaborato.

## 2.1 Introduzione al machine learning

Il **machine learning**, tradotto letteralmente "l'apprendimento di una macchina", è una sezione dell'intelligenza artificiale che consiste nell'utilizzare algoritmi che permettano ad un calcolatore di eseguire determinate azioni senza che questo sia stato esplicitamente programmato per farle; questo risultato è il frutto di un vero e proprio apprendimento che la macchina compie seguendo determinate indicazioni. Esistono vari tipi di apprendimento e solitamente questo viene selezionato analizzando il problema che si vuole risolvere:

- Apprendimento supervisionato: approccio utilizzato quando si devono affrontare problemi di classificazione o di regressione, consiste nel fornire all'algoritmo di apprendimento dati etichettati per permettergli di analizzarli e raggrupparli basandosi su caratteristiche comuni (classificazione) oppure identificare una relazione tra la variabile di output e gli input (regressione);

- Apprendimento non supervisionato: metodo selezionato quando si ha a che fare con dati non etichettati, permette di individuare i tratti comuni tra questi e creare delle aggregazioni che possono diventare fonte di analisi. Esistono varie forme di questo apprendimento come il clustering, che consiste nel raggruppare dati simili ad altri dati, e la riduzione dimensionale, dove si mantengono i tratti importanti dei dati forniti eliminando quelli considerati più "rumore" che caratteristica;

- Apprendimento di rinforzo: tecnica adottata quando si ha a che fare con un ambiente dinamico, il quale continuo cambiamento impedisce il calcolo pregresso di una possibile soluzione; in questo caso, viene utilizzata una ricompensa per incoraggiare i comportamenti corretti del modello unita ad un approccio randomico per permettere l'esplorazione dell'ambiente.

## 2.2 Deep learning

Una componente del machine learning che ha suscitato molto interesse è stata ed è sicuramente il deep learning (apprendimento profondo): questo particolare tipo di apprendimento utilizza una struttura che cerca di imitare la metodologia utilizzata dal cervello per imparare, nella quale l'input viene elaborato da una serie di strati che ne estraggono informazioni di sempre più alto livello. Questo tipo di struttura è conosciuta con il nome di "rete neurale".

### 2.2.1 Pietre miliari nella Computer vision

Nonostante il primo algoritmo funzionante basato su una rete neurale fu pubblicato da Alexey Ivakhnenko e da Lapa nel 1967, e l'applicazione all'elaborazione delle immagini da parte degli elaboratori (computer vision) vide le sue prime pubblicazioni da li a qualche anno (1980, Neocognitron proposto da Kunihiko Fukushima), non fu prima del 2011 che il deep learning venne riconosciuto come importante passo avanti in questo ambito: grazie infatti ad un'ottimizzazione nell'implementazione delle reti neurali sulle GPU, in quell'anno una soluzione (basata sul deep learning) vinse la competizione ICDAR che vedeva come soggetto gli ideogrammi cinesi, e l'anno successivo vinse il contest della ISBI sulla segmentazione delle immagini.
Grazie a questi e ad altri risultati, nel Giugno del 2012 finalmente il deep learning ottenne l'attenzione che meritava grazie ad un paper presentato da Ciresan et al. durante la conferenza principale della CVPR, il quale mostrava come le reti neurali convoluzionali max-pooling potessero superare in maniera significativa benchmark riconosciuti in precedenza come ottimi.

### 2.2.2 Reti neurali

Quando si parla di deep learing è inevitabile citare anche le reti neurali, considerando che queste fungono da struttura portante di questa tipologia di algoritmi.
Una rete neurale è un sistema computazionale ispirato al funzionamento biologico del cervello animale: questo è in grado di imparare progressivamente a svolgere delle attività, come riconoscere degli oggetti, attraverso l'analisi di esempi che gli permettano di capire la differenza tra il giusto e lo sbagliato.
Questo adattamento è il risultato di connessioni presenti tra unità di elaborazione chiamate "neuroni": ognuna di queste connessioni, che prendono il nome di "sinapsi", è la causa del passaggio di informazioni tra neuroni; tipicamente un neurone ha uno stato, rappresentato da un numero compreso tra 0 e 1, che, insieme al segnale trasmesso dalle sinapsi in entrata, determina il segnale che verrà inoltrato ai neuroni successivi.
Ciò che varia durante l'apprendimento è una caratteristica delle sinapsi, ovvero il "peso": esso determina l'importanza della sinapsi all'interno dell'algoritmo; vi sono vari metodi che fungono da apprendimento, uno dei più diffusi è sicuramente il backpropagation, che consiste nel determinare il gradiente della funzione di loss (funzione che calcola quanto la predizione restituita dalla rete sia errata rispetto al risultato corretto) rispetto al peso di ogni sinapsi presente nella rete: la possibilità che ne deriva è quella di un aggiornamento dei pesi per "correggere il tiro" nei successivi cicli di apprendimento.
I neuroni normalmente sono organizzati in strati, responsabili del modo in cui il dato in entrata viene elaborato (strati diversi possono svolgere diverse trasformazioni): il flusso dell'intera rete è dettato dalla presenza dello strato di input, strato di partenza, e dello strato di output dal quale la rete restituisce il risultato.

#### Reti siamesi

Nel 1994 venne proposta una struttura di rete neurale specializzata nell'identificare i pattern in comune negli input che le venivano presentati: si trattava della prima soluzione basata su un concetto di lavoro parallelo, ovvero la prima rete siamese.
Questo tipo di struttura presenta due reti neurali identiche, entrambe in grado di esaminare le caratteristiche dell'input che le si presenta: entrambe sono reti feed-forward (le informazioni quindi viaggiano dallo strato di input a quello di output) e implementano la backpropagation come tecnica di aggiornamento dei pesi durante l'apprendimento. L'analisi viene svolta o in parallelo per poi confrontare i due risultati al termine dell'analisi, oppure utilizzando uno dei due input come base per poi analizzare il secondo.

## 2.3 Metric learning

Considerando il target di questa tesi, una sezione a se stante la merita sicuramente il metric learning.
Molti degli approcci visto fino ad ora nell'ambito del machine learning necessitano di una "distanza" tra i vari dati per poter giungere ad una conclusione: molte volte questa viene scelta tra un set di distanza "standard" usate in questo ambito, ad esempio la distanza euclidea o la distanza coseno.
Il problema che deriva da questa scelta è che risulta molto difficile trovare una distanza che soddisfi contemporaneamente diverse tipologie di dato, quindi scegliendone una standard si generalizza una parte del processo dell'apprendimento che necessiterebbe di molta più attenzione.
A seconda del tipo di supervisione che ho a disposizione sui dati in input, il metric learning si comporta in maniera differente:

- Apprendimento supervisionato: come descritto prima, ho a disposizione un'etichetta per ogni dato. Generalizzando, l'obiettivo è avvicinare i dati con la stessa eitchetta e allontanare quelli etichettati diversamente;

- Apprendimento debolmente supervisionato: in questo caso l'algoritmo di apprendimento ha accesso ad un'etichettatura a livello di tuple (coppie, triplette). L'obiettivo diventa identificare una distanza che sia in grado di avvicinare le coppie "positive" e allontanare quelle "negative".

### 2.3.1 Funzioni di Loss

Il punto di forza del metric learning quindi è accentuare o minimizzare le differenze tra i vari elementi presenti nell'input, in modo da arrivare a ritenerne uguali alcuni e categorizzarne altri come diversi: l'aspetto quindi principale che determina questa categorizzazione è la funzione di loss, ovvero la funzione che restituisce la "bontà" della predizione della rete.

- ***Contrastive loss***[3]: per eseguire il compito sopra descritto, la contrastive loss utilizza la distanza tra le features passategli come input, distanza che normalmente è quella Euclidea, e calcola:

$$L = [d_p - m_{pos}] + [m_{neg} - d_n]$$

  dove $m_{pos}$ è il margine sotto il quale le coppie "positive" contribuiranno al calcolo della loss e $d_p$ la distanza di queste, $m_{neg}$ è il margine per le coppie "negative" e $d_n$ la loro distanza.
  Nonostante la sua semplice comprensione, questa loss non tiene conto di tutte le scale di dissimilarità presentate dagli elementi "negativi": questo aspetto limita lo spazio in cui sono rappresentate le caratteristiche delle immagini, rendendolo un sottospazio del reale insieme che comprende tutte le possibili sfaccettature;

- ***Triplete loss***[10]: loss che basa il suo funzionamento sull'analisi di triplette di dati, dove uno è utilizzato come ancora mentre i restanti rappresentano rispettivamente un set di caratteristiche etichettato come l'ancora e uno etichettato in modo differente; l'obiettivo è quello di modificare il modello di modo che la distanza tra i dati che presentano la stessa etichetta sia inferiore a quella presente tra l'ancora e il dato che presenta etichetta diversa, almeno di un margine *alpha*:

$$L = \frac{3}{2m} \sum_{i}^{m/3} [D^2_{ia,ip} - D^2_{ia,in} + \alpha]$$

dove $D_{ia,ip}$ è la distanza tra l'ancora è il dato "positivo" (ovvero con la stessa etichetta), $D_{ia,in}$ la distanza con il dato "negativo", $\alpha$ il margine tra coppie positive e coppie negative e $m$ il numero di coppie prese in considerazione nel batch. Dato che il margine non si basa direttamente sulla distanza diretta tra una coppia di dati ma sulla distanza relativa tra la coppia e un'ancora, il limite presente nella contrastive viene meno, pagando però il prezzo sotto forma di peso computazionale maggiore;

- ***Lifted Structure Loss***[7]: loss che cerca di sfruttare tutte le informazioni presenti nel batch elaborato dal modello, dove per batch si intende un estratto dell'intero dataset; in questo modo, a differenza della triplete loss, non limita il confronto con dati "negativi" al solo dato presente nella tripletta, ma ne estende il concetto a tutto il batch, permettendo una convergenza dell'apprendimento della rete migliore e più performante. Il calcolo è definito da:

$$L = \frac{1}{2|P|} \sum_{(i,j) \in P} max(0, J_{i,j})^2,$$

$$J_{i,j} = max(max_{(i,k) \in N} \alpha - D_{i,k}, max_{(j,l) \in N} \alpha - D_{j,l}) + D_{i,j}$$

Dove P ed N rappresentano rispettivamente il set delle coppie positive e negative nel batch di training, $D_{x,y}$ rappresenta la distanza tra la coppia composta da x e y mentre $\alpha$ rappresenta un margine per determinare se la coppia avrà influenza nel calcolo della loss.

- ***Angular loss***[6]: in contrapposizione ai metodi precedentemente elencati che propongono una modifica della distanza assoluta o relativa, questa loss propone una terza caratteristica da tenere in conto, ovvero l'angolo "negativo" del triangolo formato dalla tripletta utilizzata anche nella triplet loss; in questo modo si allontana il dato etichettato diversamente e si avvicinano contemporaneamente gli altri due:

$$L = [||x_a - x_p||^2 - 4tan^2\alpha||x_n - x_c||^2]$$

Dove $x_a$ è l'elemento ancora, $x_p$ l'elemento positivo, $x_n$ l'elemento negativo, $x_c$ l'approssimazione tra l'elemento positivo e l'ancora ($= \frac{x_a+x_p}{2}$) e $\alpha$ è un paramentro predefinito $> 0$ che rappresenta l'angolo limite.

- ***Margin loss***[8]: loss che nasce come estensione della contrastive, punta a rilassare il vincolo che impone alle coppie costituite da stessa etichetta (quindi coppie positive) di essere il più vicino possibile, obbligandole invece a mantenere un minimo di distanza: in questo modo la loss risulta più robusta e, utilizzando una regressione isotonica, non viene valutata la distanza assoluta ma una relativa sull'ordine delle coppie.

$$L = (\alpha + y_{ij}(D_{ij} - \beta))$$

Dove $D_{ij}$ è la distanza tra gli elementi, $\beta$ è una variabile che determina il confine tra coppia "positiva" e coppia "negativa", $\alpha$ controlla il margine di separazione tra queste e $y_{ij} \in (-1, 1)$.

- ***NCA*** (Neighbourhood Components Analysis) [9]: questa loss nasce dallo studio delle limitazioni dell'algoritmo di KNN, il quale consiste nell'etichettare un dato considerando la classificazione dei K vicini più stretti, che possono essere identificati nel costo computazionale e nella definizione di vicinanza che si utilizza. Questa loss propone un metodo diverso di selezione dei vicini: ogni vicino viene scelto tramite una funzione di probabilità $p_{i,j}$ (definita applicando una funzione softmax ad una distanza euclidea) che, a sua volta, determinerà la selezione del prossimo vicino; tramite questa selezione stocastica, è possibile calcola la probabilità che il punto scelto come vicino venga classificato correttamente. La NCA si basa sul massimizzare questa probabilità attraverso l'utilizzo di un ottimizzatore.

$$L = -\sum_i \sum_{j \in C_i} p_{ij},$$

$$p_{ij} = \frac{exp(-||Ax_i - Ax_j||^2)}{\sum_{k \neq i} exp(-||Ax_i - Ax_k||)}$$

Dove $i$ è un elemento del set di training e $j$ l'elemento che andrà a selezionare con una probabilità $p_{ij}$ (inversamente proporzionale alla distanza euclidea $d_{ij}$ nello spazio trasformato): questa probabilità viene calcolata utilizzando "A", trasformazione lineare dello spazio di input calcolata di modo che
$d_{x1,x2} = (Ax1 - Ax2)^T Q(Ax1 - Ax2)$.

# Capitolo 3

# 3 Stato dell'arte

L'obiettivo di questo capitolo è analizzare i lavori presenti nello stato dell'arte che trattano lo stesso argomento di questa tesi, ovvero il riconoscimento di firme false tramite l'utilizzo di una struttura che sfrutta il deep learning, per evidenziarne esperimenti eseguiti e risultati ottenuti.
Come riferimento ho scelto i seguenti paper perchè entrambi accennano al problema del calo delle prestazioni del loro modello quando questo viene allenato su un set in una determinata lingua e testato su un set in una lingua diversa, argomento di rilievo in questa tesi.

## 3.1 Signet: Convolutional siamese network for writer indipendent offline signature verification [3]

### 3.1.1 Introduzione

La firma ha rappresentato per anni lo strumento per eccellenza per verificare entità differenti collegate all'uomo. Di conseguenza, la verifica di questo strumento è un aspetto critico che ha visto negli anni un interesse sempre maggiore nell'eliminare l'incertezza presente nella verifica manuale, facendone un punto di ricerca per il machine learning e il riconoscimento di pattern comuni.
A seconda dell'input possiamo avere una verifica online e una offline:

- online: necessita di apparecchi appositi per l'acquisizione della firma, e permette la registrazione di tratti quali pressione della penna e velocità di scrittura;

- offline: tipicamente acquisita da scansioni della firma cartacea, restituisce solo la firma in due dimensioni.

Solitamente la firma online presenta una più precisa verifica ma presuppone l'utilizzo di hardware e software aggiuntivi. Vi sono molti casi in cui la controparte offline è l'unica opzione disponibile, sia questo dovuto ad una mancanza o ad un obbligo portato dal tipo di documento che si vuole analizzare (come assegni o contratti cartacei): per questa maggiore versatilità, questo paper si focalizza sul più difficile task del riconoscimento automatico per firme offline.

### Preprocessing

Dato che solitamente le immagini proposte in un batch hanno tutte la stessa dimensione ma le immagini presenti nei dataset presi in considerazione presentano un range che va da 153x258 a 819x1137, si è deciso di uniformare il tutto ridimensionando le immagini in entrata nel batch come 155x220 utilizzando l'interpolazione bilineare; dopodichè, l'immagine viene invertita per far valere i pixel in background 0 e normalizzata dividendo il valore di ogni pixel per la deviazione standard dei pixel delle immagini nel dataset di appartenenza.

## 3.1.2 Architettura proposta

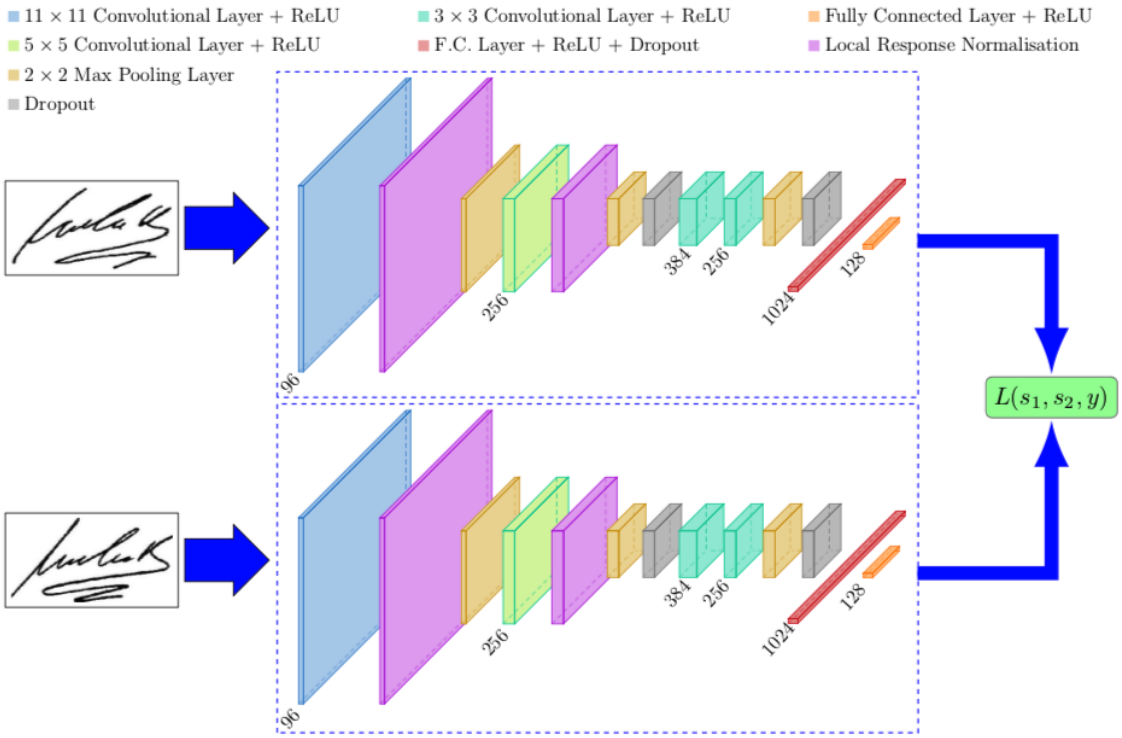


Figura 3.1: *Struttura rete* [3]

La struttura è ispirata all'architettura proposta da Krizhevsky et al. [4] per un problema di riconoscimento delle immagini. Dalla figura è possibile notare le varie tipologie di layer utilizzate:

- layer convoluzionali: insieme di neuroni con la funzione di analizzare l'immagine passata in input per estrarne le features (caratteristiche);
- layer di max pooling: layer solitamente posti dopo i convoluzionali per ridurre la dimensionalità, in questo caso data dai pixel dell'immagine; questa operazione viene svolta attraverso l'uso di un "filtro" di dimensioni HxW che seleziona al suo interno il pixel più significativo riportando unicamente quello nell'immagine di output, eliminando gli altri;

- dropout: layer utilizzato per evitare overfitting, ovvero il pericolo che la rete impari troppo dal set di apprendimento performando male quando nuovi casi, esterni a questo, le vengono proposti; questo comportamento si ottiene ponendo alcune delle unità in input (in questo caso i pixel) a 0, di modo che non abbiamo impatto sull'apprendimento del modello;

- local respose normalization: layer che non impatta sull'apprendimento con l'obiettivo di normalizzare i pixel tenendo conto di un'area quadrata intorno a lui; il motivo di questa normalizzazione risiede nella volontà di incitare l'"inibizione laterale", (fenomeno neurologico in cui un neurone eccitato sarebbe in grado di inibire l'azione dei neuroni a lui vicini) di modo che il neurone che risulta eccitato risulti un elemento di contrasto per i neuroni adiacenti provocando una reazione nel layer successivo;

- fully connected layer: stato che ha il compito di comprimere l'input derivante dal layer precedente in un singolo vettore, dove ogni elemento rappresenta la probabilità che una features appartenga ad una classe rispetto all'altra.

Dallo schema possiamo notare come sia stata utilizzata la Rectified Linear Units (ReLu) come funzione di attivazione.

**Loss function**

Per questa architettura si è scelto di utilizzare la contrastive loss strutturata in questo modo:

$$L(s_1, s_2, y) = \alpha(1-y)D_w^2 + \beta y max(0, m - D_w)^2$$

dove $s_1$ e $s_2$ sono due elementi (in questo caso immagini di firme), y è un valore binario che denota se le due firme sono originali oppure quella passata da testare risulti falsa, $\alpha$ e $\beta$ sono due costanti ,m è il margine, che nel nostro caso ha valore 1 e $D_w$ rappresenta la distanza euclidea calcolata nello spazio delle dimensioni delle feature delle immagini.

### 3.1.3 Esperimenti e risultati

Per valutare l'algoritmo per il riconoscimento delle firme, sono stati utilizzati 4 noti dataset di firme: CEDAR, GPDS300, GPDS Synthetic e il BHSig260.
Dato che l'algoritmo è stato sviluppato per essere indipendente dalla firma, sono stati selezionati M firmatari dai K disponibili (quindi $K > M$): la rete è stata addestrata sulle firme di questi M selezionati e testata sui restanti $K - M$. In un caso reale, comunque, sarebbe molto difficile avere accesso alle firme "false" per ogni firma reale che si possiede: di conseguenza è stato eseguito un esperimento dove, invece della firma falsa, viene proposta la firma reale di un altro firmatario come "falsificazione di bassa qualità".

Dalla tabella è possibile analizzare i risultati del modello, dove i dataset usati per il training sono mostrati sulle righe mentre quelli per il testing sulle colonne: la prima cosa che salta all'occhio è che il risultato migliore si ottiene allenando e testando la rete sullo stesso dataset, il che indica che sono presenti features intrinseche oltre alla lingua o all'alfabeto, dato che CEDAR, GDPS300 e GDPS synthetic condividono questi tre elementi ma hanno risultati tra loro molto più bassi che con loro stessi.
Altro fatto identificabile dai risultati è che più il dataset è grande e vario e più il modello performerà meglio.

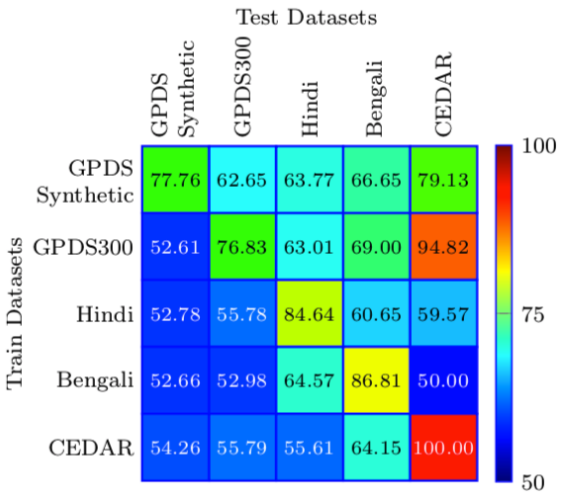


Figura 3.2: *Tabella risultati* [3]

## 3.2 Inverse Discriminative Networks for Handwritten Signature Verification [5]

### 3.2.1 Introduzione

Questo paper si focalizza sul problema della verifica delle firme, che punta a determinare se una firma è genuina o forgiata quando messa a confronto con una firma corrispondente genuina.
Nonostante le decadi passate abbiano mostrato enormi progressi in questo campo, alcune sfaccettature irrisolte la rendono una soluzione tutt'ora incompleta: per cominciare, non era presente un dataset cinese completo, il che impediva lo studio e il test della lingua cinese su questo aspetto; in secondo luogo, l'immagine rappresentante una firma fornisce informazioni sparse, alcune delle quali molto importanti (come il tratto) ma non facilmente identificabili; infine, lo stile di molti firmatari è pressochè arbitrario, il che rende molto diversa la firma originale dalla forgiata.

### 3.2.2 Architettura proposta

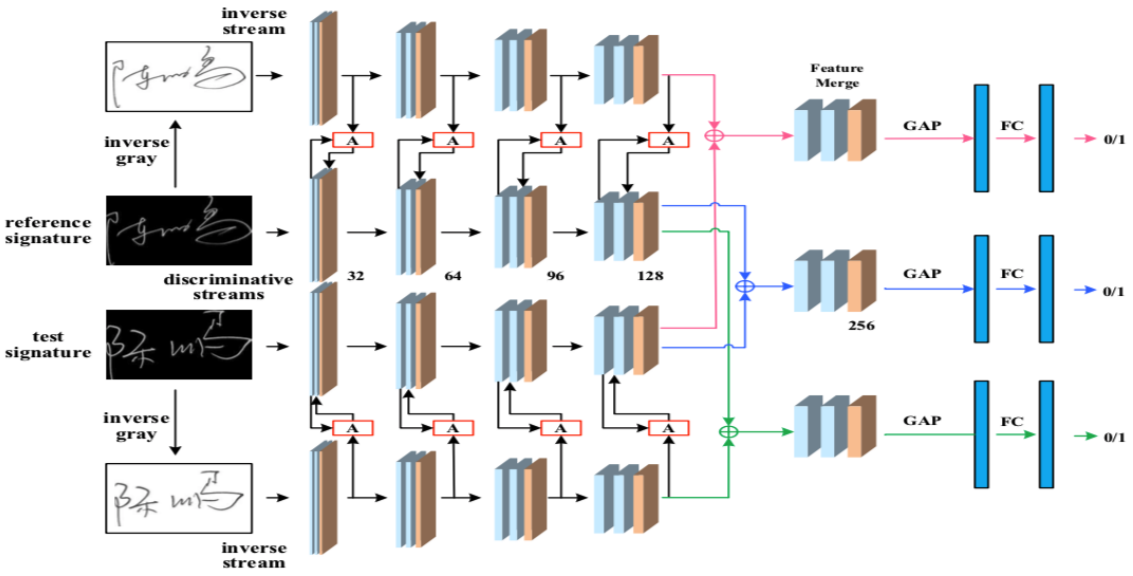


Figura 3.3: *Struttura rete* [5]

In questo elaborato, si propone una nuova tipologia di rete, la Inverse Discriminative Network (IDN): essa contiene quattro flussi pesati condivisi, due dei quali sono discriminanti mentre i rimanenti quelli inversi.
I due discriminanti ricevono rispettivamente una firma di riferimento ed una di test come input, e ne estraggono le features attraverso quattro modelli convoluzionali collegati in cascata; i due strati inversi ricevono l'immagine invertita rispetto alla scala di grigi e la firma di test.
Queste due coppie sono collegate tra loro attraverso vari moduli che inoltrano i messaggi a scale differenti per intensificare l'informazione del tratto. Le caratteristiche estratte da vari flussi discriminanti e dai flussi invertiti vengono uniti su tre diverse mappe con moduli convoluzionali, che fungono da input per tre fully connected layer per determinare la decisione.
Questo modello IDN introduce due meccanismi che mirano a risolvere le problematiche relative alla dispersione delle informazioni all'interno della firma:

- il primo è il meccanismo di supervisione invertito, che analizza l'immagine invertita (sul canale dei colori) e l'immagine di test: questa analisi dovrebbe spingere il modello a concentrarsi sul tratto, dato che una decisione presa sulla firma a colori normali dovrebbe riflettersi su quella presa a colori invertiti;

- il secondo invece è la struttura dei quattro flussi che inviano messaggi: questa caratteristica dovrebbe sottolineare quali sono i punti principali che determinano la veridicità di una firma all'interno dell'intera immagine.

**Loss function**

Considerando la struttura della rete, le decisioni prese dal modello si dovrebbero basare sul fare la stessa scelta quando viene proposta l'immagine rappresentante la firma normale e l'immagine a colori invertiti: di conseguenza, in questo elaborato viene proposta una *inverse supervision loss function* basata sull'errore della cross entropy.

### 3.2.3 Esperimenti e risultati

Per valutare l'algoritmo per il riconoscimento delle firme, sono stati utilizzati 3 dataset di firme: CEDAR, il dataset cinese creato per l'esperimento e il BHSig260. I risultati nella tabella mostrano 4 prove incrociate, questo perchè il dataset BHSig260 al suo interno contiene sia firme in Bengali sia firme in Hindi, due lingue parlate in India.

| **Train / Test** | Ours | CEDAR | BHSig-H | BHSig-B |
|---|---|---|---|---|
| Ours | **90.17** | 50.0 | 57.96 | 64.53 |
| CEDAR | 50.03 | **95.98** | 50.36 | 50.01 |
| BHSig-H | 50.0 | 50.0 | **93.04** | 74.12 |
| BHSig-B | 50.0 | 50.0 | 74.30 | **95.32** |

Figura 3.4: *Tabella dati* [5]

Dalla tabella si evince facilmente come il modello dimostri una grande robustezza quando si tratta di riconoscere le firme false quando si allena in modello su un dataset e lo si testa sullo stesso; discorso diverso si ottiene invece quando si esegue il test su un dataset in una lingua differente: possiamo notare come il risultato si aggiri intorno al 50 percento. La motivazione è legata al fatto che una firma è legata molto alla lingua di appartenenza del firmatario, analisi confermata dalla percentuale non così bassa ottenuta tra Hindi e Bengali, lingue diverse appartenenti allo stesso alfabeto.

# Capitolo 4

# 4 Modelli e risultati

In questo capitolo si presenteranno i risultati sperimentali ottenuti tramite l'applicazione al modello scelto di alcune funzioni di loss metriche.
In primis verranno elencati gli strumenti utilizzati per lo sviluppo dell'esperimento e verrà spiegata nel dettaglio la rete neurale, dopodichè verranno elencati i dataset utilizzati per gli esperimenti e mostrati i risultati ottenuti su onguno di essi.

## 4.1 Strumenti di sviluppo e framework

#### Python

Python è un linguaggio di programmazione interpretato, general-purpose di alto livello creato da Guido van Rossum nel 1991. Linguaggio object oriented, fa uso di garbage collector per mantenere le memorie pulite e permette la programmazione funzionale oltre a includere vari paradigmi, tra cui le strutture.
La filosofia di base di questo linguaggio enfatizza il codice con le indentazioni, obbligando involontariamente il programmatore a mantenere una struttura ordinata per eseguire i programmi in maniera corretta.
Python è stato associato per molto tempo all'implementazione di algoritmi di machine learning, questo per vari motivi: rispetto ad altri linguaggi di programmazione, la sintassi è più semplice e intuitiva, inoltre, insieme a R, altro linguaggio famoso nell'ambito analytics, è largamente supportata dalla comunità perchè open source. Dovesso considerare però la ragione principale, python ha accesso alle più complete librerie e framework per l'implementazione del machine learning, come tensorflow, keras e pytorch.

### Pytorch

Pytorch è una libreria open source basata su un'altra libreria per python per il machine learning ovvero Torch, sviluppata per la maggior parte dall'istituito di ricerca di Facebook.
Nonostante il suo uso principale collegato a python, ha anche un'interfaccia per C++.
Tra tutte le librerie presenti per lo sviluppo del deep learning, pytorch detiene uno dei supporti più completi per il machine learning: supporta CUDA, ovvero la piattaforma per il calcolo parallelo sviluppata da Nvidia per le sue schede grafiche e permette la computazione parallela di vari neuroni; include la differenziazione automatica, una tecnica per calcolare la derivata di una funzione definita da un computer e fornisce supporto completo alle reti convoluzionali.

### Pytorch Metric learning [11]

Elemento fondamentale di questa tesi è l'implementazione di questa libreria, sviluppata in pytorch, che permette l'utilizzo di varie caratteristiche del metric learning.
All'interno della libreria possiamo trovare:

- Distanze: funzioni che permettono di calcolare la distanza tra due embeddings (vettori di features) dati in input;

- Funzioni di loss: funzioni che permettono di calcolare il "costo" derivante dall'errore tra la predizione effettuata dalla rete e l'output corretto;

- Miners: funzioni che selezionano un subset n di embeddings all'interno di un batch; è possibile implementare miners che selezionano coppie/triplette. L'obiettivo è preselezionare gli elementi usati dalle loss all'interno del batch;

- Reducers: specificano come ridurre tutti i risultati ottenuti dalle loss calcolate sulle coppie/triplette del batch;

- Regularizers: funzioni che si applicano sui pesi e gli embeddings passati alle loss per regolarizzarne il valore.

## 4.2 Architettura utilizzata

Per l'applicazione delle varie funzioni di loss ho deciso di utilizzare il modello proposto nel paper [3], ovvero una rete neurale artificiale siamese, in cui i pesi delle sinapsi vengono condivisi tra l'estrazione delle features dell'immagine rappresentante la firma ancora e l'estrazione delle features della firma test; come optimizer, ovvero come algoritmo che calcola come ottimizzare i pesi delle sinapsi per minimizzare il risultato della loss function, è stato mantenuto l'RMSprop con un momentum rate di 0.9 ma con un learning rate di $1e^{-5}$, a differenza del paper che propone un learning rate di $1e^{-4}$, per rendere la convergenza del modello più lenta ma più precisa.

### Preprocessing

Prima di suddividere le immagini in batch da far analizzare alla rete neurale, è stato necessario uniformare le varie immagini presenti nei dataset presi in considerazione: seguendo le analisi fatte dal paper [3], tutte le immagini sono state ridimensionate a 155x220, di modo da non aggiungere rumore alle immagini più piccole e non togliere informazioni utili alle immagini più grandi; è stata mantenuta una forma rettangolare considerando che, normalmente, sui vari documenti la firma si estende orizzontalmente.
Altro aspetto importante tenuto in considerazione è il colore dell'immagine: si è deciso di riportare ogni firma in scala di grigi e a colori invertiti, di modo da non far dipendere nessuna decisione presa dal modello dal colore della penna utilizzata per scrivere sul foglio (alcuni scanner di acquisizione eseguono già questa trasformazione, ma in un caso reale non si può dare per scontato che ogni documento preso in considerazione abbia questa caratteristica) e per mettere in risalto la parte scritta rispetto al background.

### Loss

Considerando che il discorso principale degli esperimenti riguarda l'applicazione di varie loss metriche, ho deciso di dedicare una porzione dello scritto all'argomento. A livello strutturale, dopo che la rete neurale ha calcolato gli embeddings di entrambe le firme, questi vengono concatenati tramite una funzione di torch e forniti in input alla funzione di loss.
Questa funzione, facente parte della librearia PML (Pytorch Metric learning), prende come input il batch popolato da 64 vettori di features (contenenti le features delle coppie di firme) e le etichette corrispondenti, e calcola la funzione su ogni coppia/tripletta o direttamente su tutto il batch (questo dipende da che tipologia di loss si sceglie) restituendo un dizionario contenente la loss calcolata per ogni tupla: questo dizionario viene elaborato da un reducer che restituisce un valore utilizzabile dalla backpropagation per aggiornare i pesi delle sinapsi (per il mio esperimento ho scelto un reducer che calcolasse la media di ogni loss ottenuta escludendo quelle con valore nullo, ovvero 0).

### Valutazione

Per valutare l'accuratezza della rete sui dati proposti nel testing, viene eseguito un ranking sulle distanze: durante la fase di validation, iterativamente si testano tutte le threshold, ovvero i margini che dividono le due categorie a cui appartengono le firme, di modo da trovare la migliore che suddivida la classe "positiva" da quella "negativa".
Trovato il margine che divide meglio le due classi di appartenenza, vengono etichettate le due sezioni: viene calcolato il rateo dei veri positivi e dei veri negativi per poi dichiarare l'accuratezza complessiva come media di questi due valori.
Per quanto riguarda la fase di test, la threshold che ha decretato il risultato migliore in validation viene utilizzata determinare quali firme decretare falsificate e quali invece originali.

## 4.3 Esperimenti effettuati

### 4.3.1 Dataset

#### CEDAR

Questo database contiene firme appartenenti a 55 diversi firmatari con background culturale e professionale differente. Ognuno di questi firmatari ha fornito 24 firme genuine a 20 minuti di distanza l'una dall'altra.
Ognuno dei firmatari ha tentato di imitare la firma di altre 3 persone 8 volte per avere un totale di 24 tentativi di falsificazione per ogni firmatario. Di conseguenza, CEDAR mette a disposizione 1320 firme originali e 1320 firme falsificate, tutte in scala di grigio.

#### BHSig260

Il BHSig260 contiene firme in due lingue diverse, il Bengali e l'Hindi, contando rispettivamente 100 e 160 diversi firmatari. Le 24 firme genuine di ogni persona sono state collezionate lo stesso giorno, e le 30 firme forgiate sono state create scegliendo randomicamente i firmatari e chiedendo di imitare le originali proposte dagli altri.
Questo significa che, in totale, si hanno a disposizione 2400 firme originali e 3000 firme forgiate in Bengali, mentre in Hindi 3840 originale e 4800 forgiate.
Nonostante il dataset venga completo di entrambe le lingue, ho deciso di separarlo per testare anche le performance del modello su lingue diverse appartenenti allo stesso alfabeto.

#### ICDAR2011

Database nato per la competizione ICDAR nel 2011 sul riconoscimento di firme falsificate: contiene 2 lingue appartenenti a 2 alfabeti diversi, il cinese e il tedesco.
Per quanto riguarda il cinese, sono presenti 10 firmatari e 24 firme per ogni firmatario, per un totale di 240 firme originali, mentre 12 tentativi di falsificazione per un totale di 120 firme forgiate.
Per la parte tedesca, i numeri sono più miseri, avendo a disposizione sempre 24 firme per 10 firmatari, per un totale di 240, ma solo 4 tentativi di falsificazione per firma, per un totale di 40 firme forgiate.
Come per il caso del BHSig260, anche questo dataset è stato diviso in 2 parti per scopi analitici.

### 4.3.2 Creazione dei file per training, validation e testing

Considerando che i dataset utilizzati sono composti da immagini rappresentanti una singola firma, etichettata grazie all'unione di percorso e nome, è stato necessario trovare un modo per creare dei file in grado di rappresentare una coppia di immagini contenenti firme eseguite dallo stesso firmatario; per far ciò, ho scelto di creare dei file csv (Comma-Separated Values) con questo schema:

- immagine1, immagine2, etichetta
- immagine1, immagine3, etichetta

dove immagine1 e immagine2 rappresentano firme autentiche appartenenti allo stesso firmatario, che per chiarezza chiameremo firmatario A, mentre immagine3 contiene una firma sempre appartenente al firmatario A ma eseguita da un firmatario B, quindi falsa.
Questo CSV è il risultato di un programma associato ad ogni dataset che, per ogni riga, seleziona randomicamente un firmatario per il quale cerca 2 firme: se queste sono entrambe autentiche, l'etichetta sarà 0 mentre se una è un falso l'etichetta sarà 0; questo processo va avanti fino a che è possibile creare righe del CSV diverse dalle precedenti, per evitare righe doppie.
Una volta terminata la creazione, il file viene "tagliato" in 3 parti:

- 60% per il training;
- 20% validation;
- 20% test.

| Dataset | Righe totali | Righe training | Righe validation | Righe test |
|---|---|---|---|---|
| CEDAR | 8000 | 4800 | 1600 | 1600 |
| BHSig-B | 8000 | 4800 | 1600 | 1600 |
| BHSig-H | 8000 | 4800 | 1600 | 1600 |
| ICDAR-C | 5000 | 3000 | 1000 | 1000 |
| ICDAR-D | 4000 | 2400 | 800 | 800 |

Figura 4.1: *Dimensione dataset e split relativo*

Ogni esperimento è stato eseguito utilizzando il file csv di training per l'apprendimento e il file csv di validation appartenente allo stesso dataset, mentre il file csv di test è di volta in volta selezionato a seconda del dataset sul quale si vuole testare il modello.
I valori che saranno presentati nel capitolo successivo sono tutti valori ottenuti prendendo in considerazione le firme riconosciute (autentiche e false) all'interno del csv di test dei corrispettivi dataset.

## 4.4 Risultati ottenuti

In questo elaborato ho testato il modello provando ogni possibile incrocio tra i 5 dataset selezionati e le 5 loss scelte per l'apprendimento, utilizzando e confrontando l'accuratezza dopo 1, 5 e 10 epoche di training.

Considerando che il focus principale della tesi è il confronto dei risultati derivanti dall'applicazione delle varie loss quando il modello viene testato sia sullo stesso dataset di training sia su altri rappresentanti lingue e/o alfabeti diversi, ho deciso di rappresentare in una prima parte quanto la loss permetta alla rete di "apprendere" attraverso varie epoche le caratteristiche del dataset mentre in una seconda quanto, considerando il modello ottenuto dalla prima parte, questa riesca a identificare le firme false su dataset diversi da quello da cui ha imparato.

### CEDAR

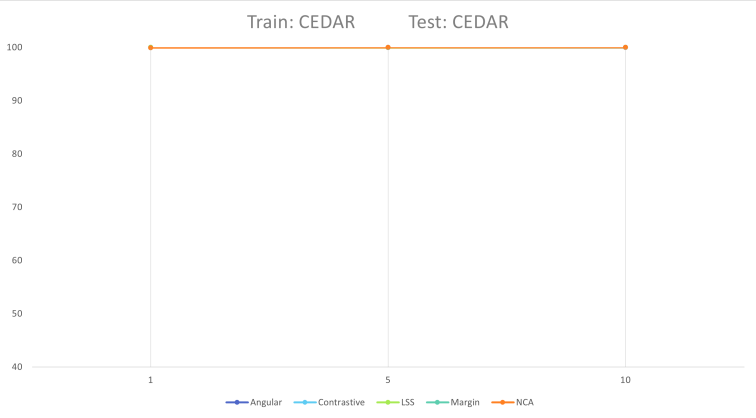


Figura 4.2: *Risultati train e test sullo stesso dataset*

Il discorso che riguarda l'apprendimento della rete attraverso le 5 diverse funzioni di loss sul datset CEDAR è di breve durata: viene largamente evidenziato infatti come questo sia un dataset nel quale le firme originali e le firme falsificate presentano differenze caratteriali semplici da identificare per la rete, a prescindere dalla loss applicata, dato che già dopo un'epoca tutte raggiungono il 100% di accuratezza sulla parte riservata al test.


Figura 4.3: *Risultati cross dataset*

Nei 4 grafici appena proposti, invece, si possono osservare i risultati della rete quando questa viene allenata attraverso il CEDAR ma testata sugli altri 4 dataset scelti. Osservando i grafici nel loro complesso si può facilmente notare come l'*Angular loss* permetta al modello di enfatizzare le caratteristiche delle firme falsificate evidenziate imparando da se stesso e di estenderle al dataset BHSig-B e al BHSig-H, rispettivamente contenenti firme in lingua bengalese e indiana, a differenza delle altre loss testate.
Unica loss che si scosta dal risultato delle altre, superando anche (dopo 5 epoche) il risultato ottenuto dall'*angular* stessa, è la *contrastive* loss sul test effettuato con il dataset ICDAR-D, dataset in lingua tedesca, che riesce nelle epoche dalla $1^a$ alla $5^a$ ad associare le differenze tra firme originali e le falsificate tra i due dataset probabilmente utilizzando come elemento l'alfabeto in comune.

### BHSig-B

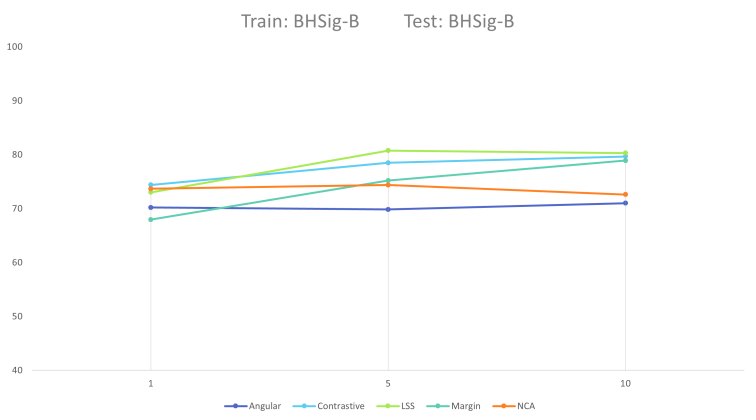


Figura 4.4: *Risultati train e test sullo stesso dataset*

Il grafico riporta i risultati delle 5 loss metriche nel caso del single-domain, ovvero training set e testing set appartenenti al BHSig-B: si può osservare come la *lifted structure* e la *contrastive* siano le funzioni che non solo riescono ad ottenere i risultati migliori ma contemporaneamente riescono ad imparare, nel corso delle epoche, a distinguere sempre meglio firme autentiche da firme falsificate.
Si può menzionare anche la *margin* in questo caso che si avvicina ai risultati delle precedenti ma molto più lentamente, avvicinandosi ai risultati solo dopo 10 epoche.
In controtendenza rispetto a tutte troviamo la *NCA* che dopo la prima epoca ottiene risultati simili alle loss sopracitate ma che, più si prosegue con le epoche, più perde accuratezza sul dataset di test quasi come se, invece di permettere al modello di "imparare" dal training, lo allontanasse da quelle caratteristiche da analizzare per ottenere risultati più alti.

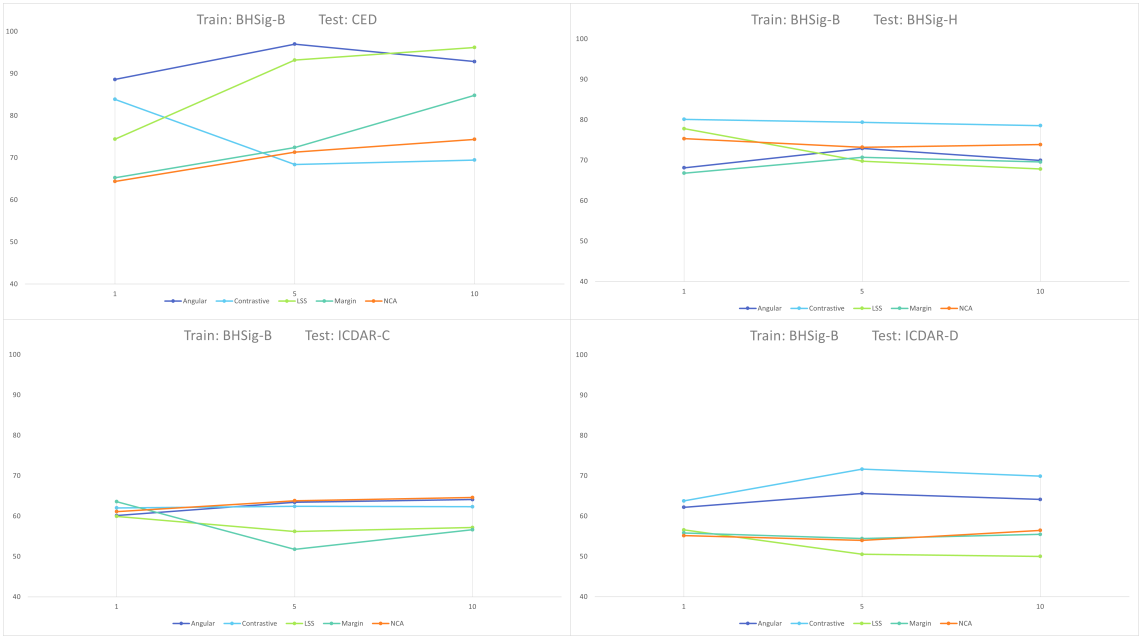


Figura 4.5: *Risultati cross dataset*

Per quanto riguarda il paragrafo cross-domain del BHSig-B, i grafici risultano molto diversi rispetto a quelli esaminati prima nel caso del CEDAR.
Ad una prima occhiata si può subito notare come il modello con le varie loss non rispetti un comportamento costante su tutti i dataset di test, ma che ottenga risultati diversi a seconda della complessità delle firme false (rispetto alle originali) e delle caratteristiche che il dataset di test presenta in comune con quello di training.
Sicuramente l'*angular* può essere ritenuta come la loss tramite la quale si osservano i risultati migliori: nel CEDAR si sfiora il 100% dopo la $5^a$ epoca, nel ICDAR-D e nel ICDAR-C nelle prime 5 epoche si può apprezzare un aumento dell'accuratezza nonostante i dataset non presentino ne lingua ne alfabeto come elemento comune mentre nel BHSig-H (alfabeto comune) si supera il 70% di accuratezza.
Allo stesso livello dell'*angular* troviamo sicuramente la *contrastive*, tramite cui il modello testato sul BHSig-H si tiene sopra l'80% mantenendo il rank maggiore, mentre nel caso del dataset tedesco (ICDAR-D), con il quale non ha nulla in comune, supera ogni altra loss ottenendo un risultato superiore al 70%.

### BHSig-H


Figura 4.6: *Risultati train e test sullo stesso dataset*

Confrontato con gli altri test single-domain, il modello sul BHSig-H presenta risultati singolari: non importa la loss che viene applicata, questo manterrà un andamento costante attraverso tutte e 10 le epoche di training, presentando risultati diversi nel caso in cui questi siano comparsi già dalla prima epoca.
Nel dettaglio, la *LSS* sembra essere la loss tramite la quale si ottenga l'accuratezza maggiore, di poco superiore all'80%; la *contrastive* e la 1*NCA* sembrano tenere il passo, discostandosi solo di qualche punto percentuale.

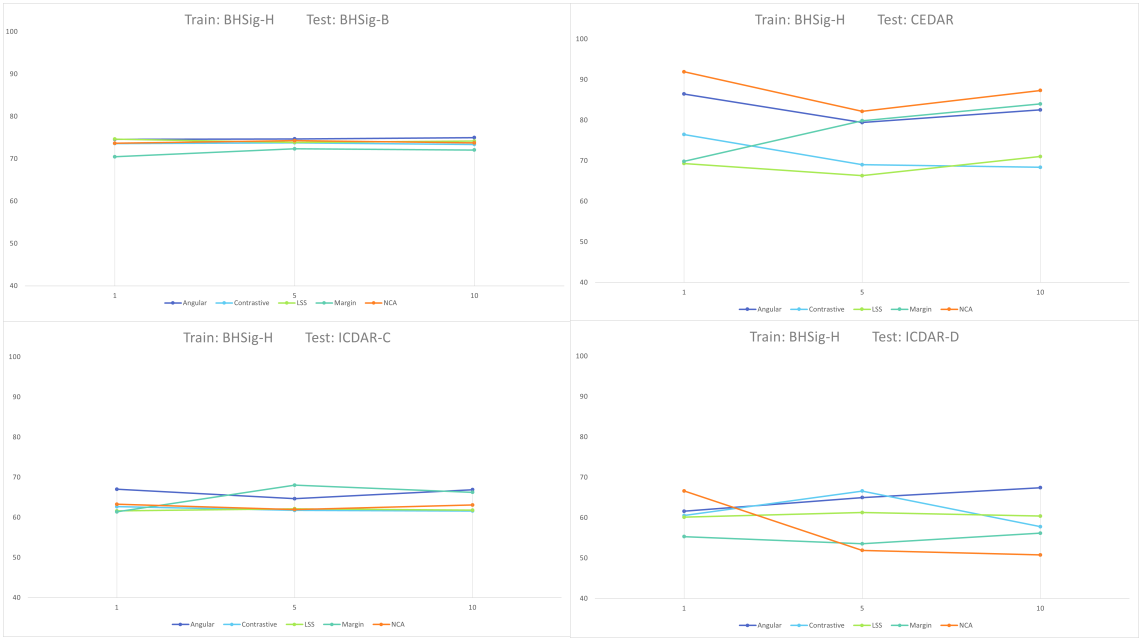


Figura 4.7: *Risultati cross dataset*

Sicuramente di maggiore interesse invece sono i rusultati ottenuti nei test sugli altri dataset: sull'unico che condivide un elemento comune, ovvero il BHSig-B, il comportamento adottato dal modello sembra essere molto simile a quello riscontrato nel test single-domain.
Diverso discorso si deve fare per i dataset con lingua e alfabeto diversi, dove risulta necessario separare l'analisi tra CEDAR e ICDAR: nel primo caso, la *NCA* e l'*angular* sembrano mantenere la posizione migliore di ranking sulla prima epoca, dopo la quale la *margin*, con comportamento totalmente opposto (ovvero imparando invece di perdere nozioni al progredire delle epoche), raggiunge i risultati presentati da queste.
Nel secondo caso invece, sulla parte in lingua cinese, la *contrastive* e l'*angular* si distaccano totalmente dalle altre mantenendo una distanza di quasi 10 punti percentuali mentre sulla parte in tedesco i potrebbe fare lo stesso discorso si mantenessero i risultati ottenuti dopo 5 epoche, dato che sulla $10^a$ la *contrastive* presenta un netto calo di prestazioni.

### ICDAR-C

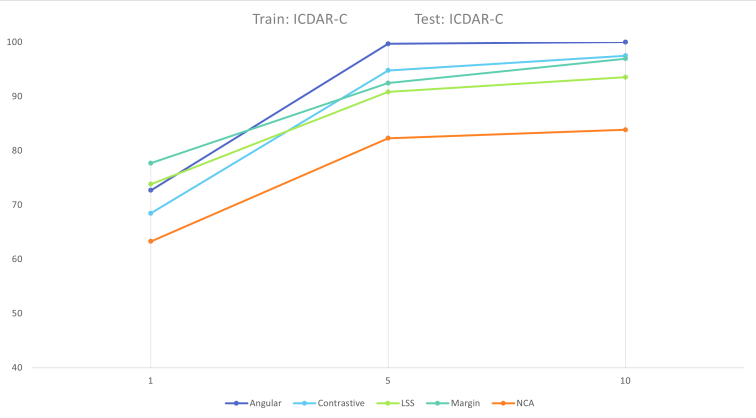


Figura 4.8: *Risultati train e test sullo stesso dataset*

Nella famiglia dei dataset dell'ICDAR, ovvero l'ICDAR-C e l'ICDAR-D, si può facilmente notare dai grafici rappresentanti i test single-domain il fenomeno dell'overfitting: questo comportamento si verifica quando, come nel nostro caso, non si ha a disposizione una gran varietà di dati nel dataset, permettendo ad un modello più complesso del necessario di imparare ogni singola differenza tra tutte le firme presenti e, di conseguenza, di ottenere dopo un numero sufficiente di epoche risultati vicini al 100%.

Data questa premessa, è facilmente identificabile questo fenomeno su tutte e 5 le loss applicate, anche se i risultati ottenuti sono diversi: da sottolineare come l'*angular* riesca a raggiungere il 100% dopo sole 5 epoche, a differenza delle altre che, nonostante risultati migliori dopo la prima, presentano un apprendimento molto più lento.

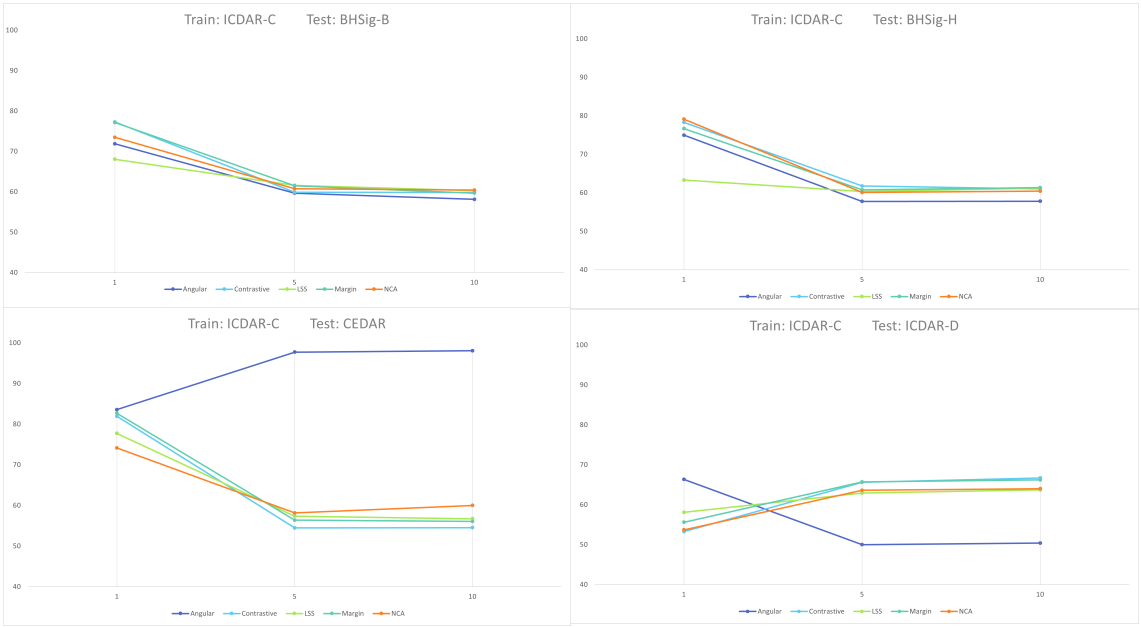


Figura 4.9: *Risultati cross dataset*

Come si presenta nel test single-domain, l'overfitting è facilmente delineabile anche nei test eseguiti sugli altri dataset: di interessante analisi sono 2 casi particolari:

- il comportamento dell'*angular* loss sul CEDAR, che segue esattamente il tracciato percorso nel test single-dataset con overfitting;
- il comportamento sempre dell'*angular* sul dataset tedesco, ovvero l'ICDAR-D, che non solo va controcorrente rispetto a tutte le altre loss facendo perdere accuratezza al modello, ma subisce un calo di 20 punti percentuali dopo la $1^a$ epoca.

### ICDAR-D

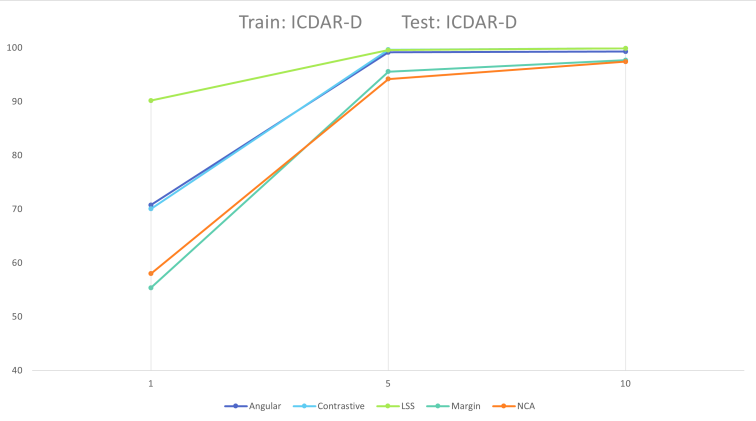


Figura 4.10: *Risultati train e test sullo stesso dataset*

Come anticipato, anche in questo caso è presente il fenomeno dell'overfitting, affrontato in maniera diversa dalle varie loss rispetto al caso del ICDAR-C: possiamo notare come la *LSS* dimostri di identificare da subito le caratteristiche delle firme presenti superando il 90% già dalla prima epoca di training, mentre *angular* e *contrastive* seguono a 20 punti percentuali recuperati solo dopo la $5^a$.

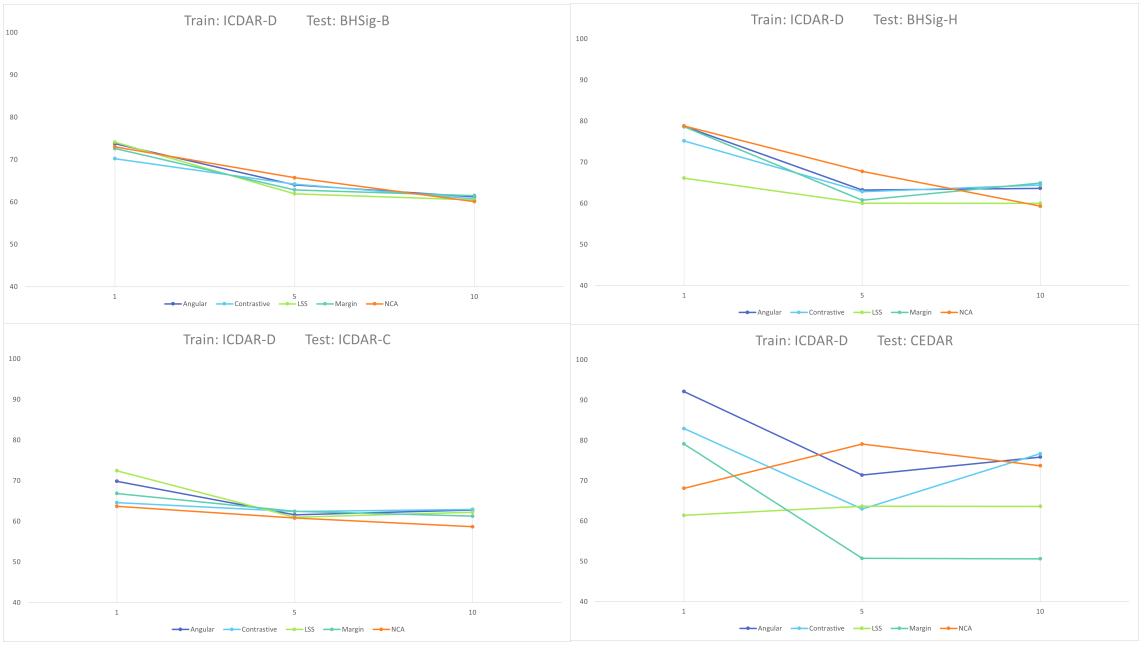


Figura 4.11: *Risultati cross dataset*

Nei test cross-domain del ICDAR-D si escludono già ad una prima occhiata i casi particolari presenti invece nell'altro dataset della famiglia ICDAR: il comportamento del modello con le varie loss mantiene un andamento costante negativo tra la $1^a$ e la $15^a$ epoca su tutti e 4 i dataset presentati come test, tranne per un solo segmento, quello tracciato dalla *NCA* sul CEDAR che invece registra un incremento nell'accuracy di circa 10 punti percentuali. Dalla $5^a$ alla $10^a$ epoca, invece, su CEDAR e BHSig-H si può notare come la *contrastive* e l'*angular* invertano la tendenza come se avessero individuato sul dataset di training delle caratteristiche comuni lasciate in disparte nelle prime 5 epoche, caratteristiche che invece permettono una generalizzazione del focus che porta risultati migliori.

### Considerazioni single-domain

Osservando tutti i casi analizzati per la categoria single-domain (ovvero utilizzando train set e test set appartenenti allo stesso dataset) si può notare come non sia possibile individuare con chiarezza la loss o la coppia di loss che meglio si presta a svolgere questo compito.
Nel caso di BHSig-B, BHSig-H e ICDAR-D la *LSS* sembra essere una delle migliori loss per individuare le caratteristiche intrinseche nel dataset, ma quando si analizzano gli score ottenuti sul ICDAR-C, risulta essere la penultima in rank (il CEDAR è escluso da questo discorso dato che l'accuratezza è 100% per ogni loss in ogni epoca); al contrario, l'*angular* ottiene i risultati peggiori su BHSig-B e BHSig-H, non si distingue nel ICDAR-D ma risulta essere l'unica loss nel dataset in lingua cinese, l 'ICDAR-C, che raggiunge il 100%.

### Considerazioni cross-domain

Escludendo i casi già delineati nel paragrafo precedente, rimangono i test effettuati cross-domain, ovvero quando il modello viene sottoposto un dataset come fonte per l'apprendimento ma in test gli viene sottoposto un altro dataset: già da questa descrizione si può comprendere come il cross-domain sia un concetto molto più complesso del corrispettivo single-domain, dato che sto chiedendo al modello di eseguire delle prove su qualcosa che ha studiato ma sotto un altro dominio, in alcuni casi totalmente diverso.
L'interesse principale che deriva però da questa analisi risiede principalmente ad un'ambiente di test che si avvicina a quello che è il mondo reale, in cui non esiste il concetto di "pre-determinato", ma governa la casualità.
Come nel caso del single-domain, non è possibile individuare una loss che, applicata al modello, gli permetta di raggiungere i migliori risultati in ogni incrocio tra dataset di train e dataset di test; è presente però una loss che permette al modello di ottenere in media un buon riscontro sui test, riscontro che sottolinea quanto già detto nella premessa di questa tesi ovvero che la funzione di loss metrica va scelta analizzando bene i dati e le casistiche che si vogliono affrontare.

**Risultati e confronto con stato dell'arte**

Di seguito, oltre che a mostrare i risultati ottenuti dal mio modello sui vari dataset di test dopo 5 epoche di training, ho messo a confronto il mio lavoro con lo stato dell'arte citato nel capitolo 3 (nella tabella ci si riferisce ai risultati del modello proposto in questa tesi con la sigla MS "Mia Soluzione"):

| Train | Test | MS(loss) | SigNet | IDN |
|---|---|---|---|---|
| CEDAR | BHSig-B | 54.12(Ang) | **64.15** | 50.01 |
| CEDAR | BHSig-H | **57.28**(Ang) | 55.61 | 50.36 |
| BHSig-B | CEDAR | **97.00**(Ang) | 50.00 | 50.00 |
| BHSig-B | BHSig-H | **79.38**(Con) | 64.57 | 74.12 |
| BHSig-H | CEDAR | **82.16**(NCA) | 59.57 | 50.00 |
| BHSig-H | BHSig-B | **74.69**(Ang) | 60.65 | 74.12 |

Dalla tabella si evince come, tolto il caso del CEDAR come train set e BHSig-B come test set, tutti gli altri risultati migliori cross domain sono stati ottenuti tramite il modello proposto in questo elaborato. Nei casi in cui il CEDAR risulta essere il dataset di test, possiamo notare come il modello ottenga un'accuratezza di molto superiore a quella riscontrata invece dai modelli presentati nello stato dell'arte.

Nella seguente tabella, invece, sono riportati i risultati ottenuti includendo anche l'ICDAR, dataset non utilizzato dagli altri elaborati citati nel capitolo 3:

| Train | Test | MS(loss) |
|---|---|---|
| CEDAR | ICDAR-C | 57.91(Ang) |
| CEDAR | ICDAR-D | 59.40(Con) |
| BHSig-B | ICDAR-C | 63.81(NCA) |
| BHSig-B | ICDAR-D | 71.66(Con) |
| BHSig-H | ICDAR-C | 68.04(Mar) |
| BHSig-H | ICDAR-D | 66.60(Con) |

(Sono stati esclusi dalla tabella i risultati cross-domain ottenuti dal ICDAR in generale dato il fenomeno dell'overfitting in training che purtroppo sfalsa l'accuratezza, come si può notare dai grafici mostarti in precedenza)

# Capitolo 5

# Conclusioni

Essendo questo il capitolo conclusivo della tesi, inizialmente verrà affrontato l'obiettivo della tesi e se attraverso i vari esperimenti è stato raggiunto.
Successivamente, verranno menzionate le future implementazioni che sarebbe possibile affrontare per migliorare o evolvere il lavoro svolto.

## 5.1 Analisi dei risultati

L'obiettivo di questa ricerca è stato valutare l'efficacia di un modello di deep learning per il riconoscimento di firme falsificate nel caso in cui i dati di addestramento provengano da un dataset con lingua e/o alfabeto diversi rispetto ai dati di test. Per raggiungerlo, ho selezionato cinque funzioni di loss appartenenti al metric learning per poi analizzarne e confrontarne i risultati ottenuti.
Come base è stata utilizzata la stessa rete neurale, quindi le differenze nell'accuratezza sono dovute soltanto all'applicazione della loss.
In conclusione, possiamo dire che la premessa fatta nel capitolo 2:

*"Il problema che deriva da questa scelta è che risulta molto difficile trovare una distanza che soddisfi contemporaneamente diverse tipologie di dato, quindi scegliendone una standard si generalizza una parte del processo dell'apprendimento che necessiterebbe di molta più attenzione"*

è stata confermata dai risultati ottenuti che, oltre a questo, dimostrano non solo l'importanza della scelta ma anche il divario tra il comportamento di un modello con la giusta loss per il compito che deve svolgere e un modello con una loss sulla quale non sono state svolte ricerche prima di essere scelta.

## 5.2 Approfondimenti futuri

Il punto che sicuramente richiede maggiore attenzione è il discorso dataset: quando si trattano modelli di intelligenza artificiale, l'aspetto più importante riguarda non solo la quantità di dati a disposizione ma anche la varietà di questi, per permettere al modello di imparare a distinguere più classi possibili e riconoscere soprattutto le varie sfaccettature che queste possono presentare. Questo elemento si riflette molto chiaramente in questo elaborato quando si tratta il dataset ICDAR nella sua completezza: questo offre una limitata varietà di firmatari, sia per quanto riguarda la lingua cinese sia per il tedesco.
I numeri proposti precedentemente parlano chiaro: con questa limitazione, il modello si adatta in fretta alle caratteristiche intrinseche delle firme presenti nella parte di training, aumentando l'accuratezza in fase di test nel caso del single-domain (dato però non totalmente attendibile per le motivazioni elencate in precedenza) ma diminuendo drasticamente la capacità di generalizzare quello che ha imparato, effetto molto visibile nei test di cross-domain.

Parlando della struttura, dai risultati ottenuti su tutti i cross-domain test, è chiaro come alcune loss siano più adatte rispetto ad altre per il nostro specifico scopo: oltre a questo aspetto, viene comunque sottolineata la variazione di efficacia di queste legata alla specifica natura del domain shift affrontato; quindi, una possibile implementazione futura potrebbe includere più metriche simultaneamente, gestite a monte da un sistema di weighting per un'adeguata implementazione in base alla necessità.

Altra grossa influenza sulle prestazioni del modello deriva dal tuning di tutte le variabili che possono influenzare direttamente o indirettamente l'andamento dell'apprendimento: alcuni esempi possono essere il *seed*, utilizzato per determinare il peso casuale delle sinapsi prima della $1^a$ epoca di training o il learning rate dell'ottimizzatore, che influisce sulla velocità con cui la rete apprende nuove caratteristiche in fase di apprendimento (questo parametro potrebbe attenuare il fenomeno dell'overfitting permettendo alla rete di ottenere score migliori anche con dataset di test diversi dal dataset di training).

# Bibliografia

[1] M. Mane (2014). *Signs of traced signature in forgery*

[2] M. Diaz, M.A. Ferrer, D. Impedovo, M.I. Malik, G. Pirlo, R. Plamondon (2019). *A Perspective Analysis of Handwritten Signature Technology*

[3] S. Key, A. Dutta, J.I. Toledo, S.K. Ghosh, J. Llados, U. Pal (2017). *SigNet: Convolutional Siamese Network for Writer Independent Offline Signature Verification*

[4] A. Krizhevsky, I. Sutskever, G. E. Hinton, *Imagenet classification with deep convolutional neural networks* in NIPS, 2012, pp. 1097–1105.

[5] P. wei, H. Li, P. Hu (2019). *Inverse Discriminative Networks for Handwritten Signature Verification*

[6] J. Wang, F. Zhou, S. Wen, X. Liu, Y. Lin (2017). *Deep Metric Learning with Angular Loss*

[7] H.O. Song, Y. Xiang, S. Jegelka, S. Savarese (2015). *Deep Metric Learning via Lifted Structured Feature Embedding*

[8] C.Y. Wu, R. Manmatha, A.J. Smola, P. Krahenbuhl (2018). *Sampling Matters in Deep Embedding Learning*

[9] J. Goldberger, S. Roweis, G. Hinton, R. Salakhutdinov (2004) *Neighbourhood Components Analysis*

[10] E. Hoffer, N, Ailon (2018). *Deep metric learning using Triple network*

[11] K. Musgrave, S. Belongie, S. N. Lim (2020). *Pytorch Metric Learning*, https://kevinmusgrave.github.io/pytorch-metric-learning/