



Università degli studi di Cagliari

FACOLTA' DI SCIENZE

Corso di Laurea Triennale in Informatica

Un Approccio per la *Domain Generalization* in Ambito della Re-identificazione

Relatore: Andrea Loddo

Candidato: Lorenzo Leuzzi – 60/61/65785

Anno Accademico 2021/2022

INDICE

CAPITOLO 1: Introduzione al problema e all'intelligenza artificiale 3

1.1 Introduzione al problema	3
1.2 Panoramica sull'intelligenza artificiale (Intelligenza Artificiale vs Machine Learning vs Deep Learning)	5
1.3 Reti neurali artificiali	6

CAPITOLO 2: Materiali e metodi9

2.1 Convolutional Neural Network (CNN)	9
2.2 Generative Adversarial Network (GAN).....	9
2.3 CycleGAN.....	11
2.4 StarGAN.....	14
2.5 Dataset impiegati	18
2.5.1 Market-1501.....	18
2.5.2 Duke-MTMC.....	19
2.5.2 MSMT17	20
2.6 Librerie usate.....	20
2.6.1 TensorFlow.....	20
2.6.2 PyTorch	21
2.7 Colab	22

CAPITOLO 3: Valutazione sperimentale23

3.1 CycleGAN per il trasferimento di stile da un dataset ad un altro	24
3.1.1 Discussione risultati.....	26
3.1.2 Re-identificazione con data augmentation	27
3.2 CamStyle.....	28
3.2.1 Discussione risultati.....	31
3.2.2 Re-identificazione con data augmentation.....	32
3.2.3 Re-identificazione cross-dataset.....	33

CAPITOLO 4: Conclusioni e sviluppi futuri37

CAPITOLO 1: Introduzione al problema e all'intelligenza artificiale

1.1 Introduzione al problema

Nonostante gli straordinari successi dell'elaborazione dell'informazione, che stanno esercitando un impatto di portata storica nella vita quotidiana, competenze percettive come per esempio localizzare un oggetto in una scena, riconoscere la voce in ordinarie condizioni reali, prendere decisioni basate sul “senso comune”, risultano ancora compiti estremamente difficili per le macchine.

Oggigiorno si investono molte risorse umane, materiali e finanziarie sulla previdenza sociale per controllare e prevenire crimini che influiscono negativamente sulla società. Un sistema di video sorveglianza intelligente può diventare una parte importante per la sicurezza degli ambienti pubblici e delle persone.

Le applicazioni dell'intelligenza artificiale in questo ambito sono diverse ma con questo elaborato ci si soffermerà in particolare sulla re-identificazione di un individuo.

Re-identificare un soggetto significa creare una corrispondenza tra diverse immagini di una stessa persona, come illustrato nella *Figura 1*. Questo implica una considerevole sfida per le intelligenze artificiali perché le immagini possono provenire da diverse camere o dalla stessa camera ma in diverse occasioni [1]. Un sistema di re-identificazione, quindi, deve saper riconoscere un individuo nonostante la differenza di posa, di risoluzione della camera, l'aggiunta o la rimozione di dettagli, e così via.



Figura 1: re-identificazione di un'immagine [1]

Quando si parla di deep learning bisogna avere a disposizione grandi quantità di dati. Ma raccogliere dati per comporre un dataset che rimanga attendibile rispetto alle variazioni di immagine del soggetto è difficile e costoso.

Una parte del problema della re-identificazione di un individuo è ottenere un dataset che attenui le disparità degli stili delle camere senza effettivamente raccogliere nuovi elementi.

Il metodo proposto è arricchire l'insieme di dati di partenza (*data augmentation*) trasferendo lo stile di diverse immagini raccolte. Si parla di un sistema di *Image-to-Image Translation*, cioè il trasferimento di un'immagine da un dominio ad un altro con l'obiettivo di far apprendere al modello la mappatura tra l'input e l'output. In questo modo si vuole offrire un sistema basato su nuove immagini, rappresentanti idealmente delle camere sintetiche, in cui si possano identificare anche individui provenienti da immagini eterogenee, in un approccio di *domain generalization*.

Lo scopo del presente lavoro di tesi è quello di realizzare immagini sintetiche tramite Generative Adversarial Network (GAN) per facilitare il compito della re-identificazione di un individuo, grazie all'investigazione di architetture di re-identificazione preesistenti, raffinate con l'utilizzo di tali immagini.

1.2 Panoramica sull'intelligenza artificiale

L'intelligenza artificiale [2] è una disciplina che studia in che modo si possono realizzare sistemi informatici intelligenti in grado di simulare la capacità e il comportamento del pensiero umano.

L'Intelligenza Artificiale (IA) è un termine generico che comprende diverse tecniche, a volte complesse, ma anche semplicemente una serie di regole o algoritmi base che ricordano l'intelletto umano. Degli esempi di IA di tipo elementare sono: l'algoritmo che permette al fantasma di inseguire *PAC-MAN* nel famoso gioco elettronico, oppure l'insieme di regole che stabiliscono la mossa successiva di un bot che gioca a *Tic-Tac Toe*.

Quando si parla di Machine Learning (ML), invece, le regole non vengono somministrate dal programmatore al computer ma è la macchina che cerca di scoprirle da sola. Quindi Machine Learning è un sottoinsieme dell'intelligenza artificiale che si occupa di creare sistemi che apprendono o migliorano le performance in base ai dati che utilizzano.

Le reti neurali, a loro volta, sono un sottoinsieme del machine learning e costituiscono l'elemento centrale degli algoritmi di Deep Learning. Il loro nome e la loro struttura sono ispirati al cervello umano, imitando il modo in cui i neuroni si inviano segnali. Da un punto di vista scientifico, si può dire che il Deep Learning sia l'apprendimento da parte delle "macchine" attraverso dati somministrati dal programmatore e grazie all'utilizzo di algoritmi prevalentemente di calcolo statistico.

La *Figura 2* mostra i sottoinsiemi dell'intelligenza artificiale.

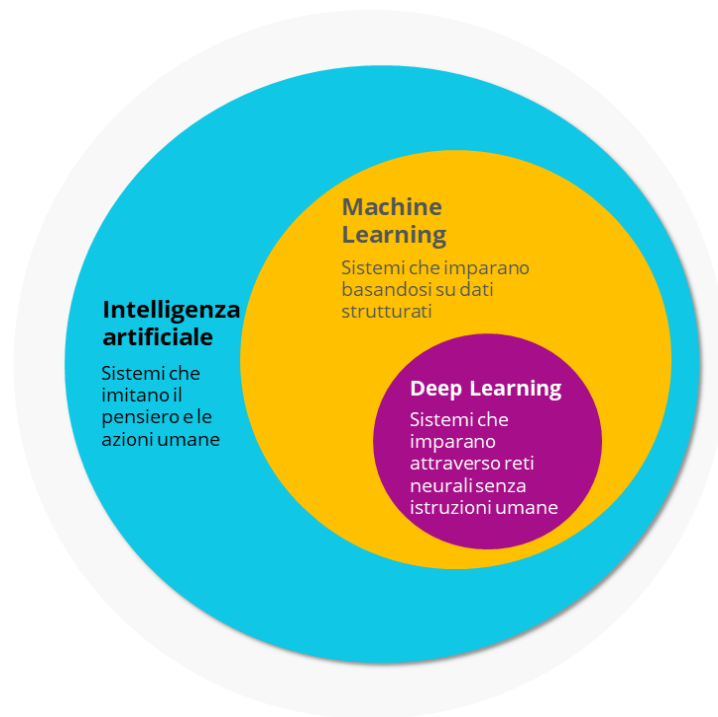


Figura 2: Intelligenza Artificiale vs Machine Learning vs Deep Learning, immagine presa da ionos.it.

1.3 Reti neurali artificiali

Si può capire il funzionamento delle reti neurali partendo da cosa sono i neuroni e dal modo in cui sono collegati tra loro.

Un neurone (o nodo) nell'ambito del deep learning è semplicemente un contenitore che include un numero tra 0 e 1 detto di attivazione. Una rete neurale artificiale [3] riceve segnali esterni su uno strato di nodi di ingresso, ciascuno dei quali è collegato con numerosi nodi interni, organizzati in più livelli. Ogni nodo elabora i segnali ricevuti e trasmette il risultato a nodi successivi fino ad arrivare allo strato finale di output (*Figura 3*).

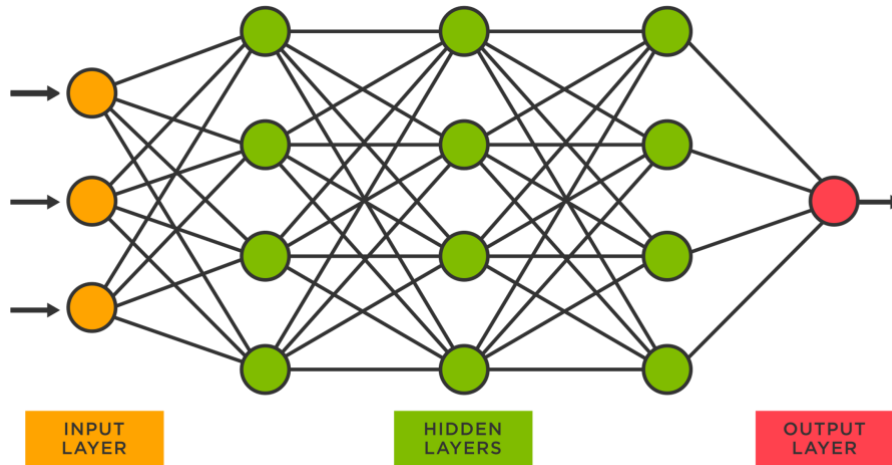


Figura 3: una rete neurale.

L'elaborazione può essere anche molto sofisticata ma in un caso semplice si può pensare che i singoli ingressi vengano moltiplicati per un opportuno valore detto peso, il risultato delle moltiplicazioni viene poi sommato ad una costante detta *bias* e se la somma supera una certa soglia il neurone si attiva attivando la sua uscita. Il peso serve a quantificarne l'importanza: un ingresso molto importante avrà un peso elevato, mentre un ingresso poco utile all'elaborazione avrà un peso inferiore.

Quindi la maniera in cui i nodi a di uno strato k determinano quelli del successivo può essere espressa come:

$$\begin{bmatrix} a_0^{k+1} \\ a_1^{k+1} \\ \vdots \\ a_n^{k+1} \end{bmatrix} = \sigma \left(\begin{bmatrix} w_{0,0} & w_{0,1} & \cdots & w_{0,m} \\ w_{1,0} & w_{1,1} & \cdots & w_{1,m} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n,0} & w_{n,0} & \cdots & w_{n,m} \end{bmatrix} \begin{bmatrix} a_0^k \\ a_1^k \\ \vdots \\ a_n^k \end{bmatrix} + \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_n \end{bmatrix} \right)$$

Scritta in maniera più compatta:

$$\mathbf{a}^{k+1} = \sigma(W\mathbf{a}^k + \mathbf{b})$$

dove $\mathbf{a}^{(k)}$ è il vettore di tutti i valori dei nodi allo strato k , W è la matrice dei pesi, \mathbf{b} il vettore dei *bias* e σ la funzione di attivazione (*Sigmoid* per esempio).

La funzionalità più interessante delle reti neurali è la possibilità di auto apprendimento che, in pratica, significa modificare i pesi delle connessioni in modo da poter predire con accuratezza il risultato corretto dell'elaborazione.

Ciò comporta la definizione di una *funzione di costo (o obiettivo)* tale che, per la soluzione ottimale, nessuna soluzione ha un costo inferiore al costo della soluzione ottimale. La funzione di costo è un concetto importante nell'apprendimento, poiché si tratta di una misura di quanto è lontana la soluzione del problema che si vuole risolvere.

Addestrare (o allenare) una rete neurale artificiale significa iterare l'auto apprendimento. Il costo di un allenamento è calcolato con il metodo dello scarto quadratico, cioè viene considerato il quadrato della differenza tra il risultato ottenuto e quello aspettato. Questo valore è basso quando la rete riesce ad ottenere risultati accurati e alto quando l'output non è affidabile. Si considera dunque la media dei costi dei vari allenamenti e si cerca di minimizzare quel valore con strumenti matematici: discesa del gradiente.

La discesa del gradiente è una tecnica che consente di determinare i punti di massimo e minimo di una funzione a più variabili. Il gradiente dà la direzione lungo la quale si ha il massimo incremento (o decremento nel caso dell'anti-gradiente). Quindi, nella fase di addestramento, variando i pesi w (parametri del modello) si può aumentare o diminuire la funzione obiettivo; la prestazione della rete sarà funzione delle variabili w , e sarà massima quando si raggiunge il minimo della funzione obiettivo, il che si ottiene applicando il metodo del gradiente e aggiornando iterativamente i valori dei pesi.

CAPITOLO 2: Materiali e metodi

2.1 Convolutional Neural Network (CNN)

Quando si parla di deep learning, una rete neurale convoluzionale (CNN o ConvNet) è un tipo di rete neurale artificiale in cui il pattern di connettività tra i neuroni si ispira all'organizzazione della corteccia visiva animale. In particolare, queste reti sono adatte per il riconoscimento di immagini.

Le reti neurali convoluzionali funzionano come tutte le reti neurali: uno strato di input, uno o più strati nascosti (che effettuano calcoli tramite funzioni di attivazione) e uno strato di output con il risultato. La differenza sono appunto le convoluzioni. Ogni strato ospita quella che viene chiamata *feature map*, ovvero la specifica caratteristica che i nodi si occupano di cercare. In parole più semplici, analizzando un'immagine, un primo stadio potrebbe essere quello di riconoscere le silhouette delle figure.

Strati successivi potrebbero per esempio riconoscere occhi, orecchie, mani, etc. Alla fine, l'ultimo strato, potrebbe essere in grado di riconoscere e distinguere animali, persone, automobili, etc.

Nella *Figura 4* è schematizzato il funzionamento della convoluzione di un'immagine.

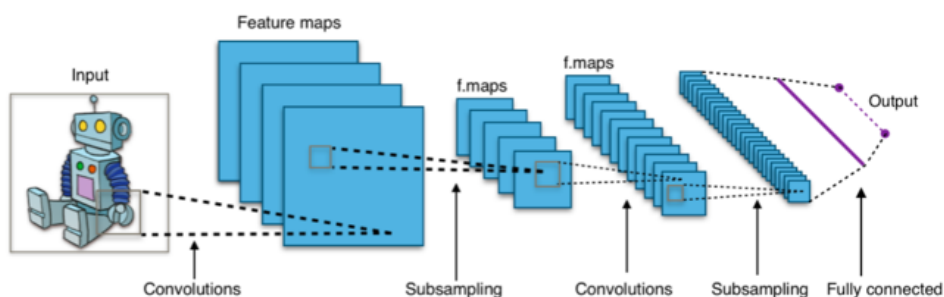


Figura 4: il processo di convoluzione.

2.2 Generative Adversarial Network (GAN)

Una Rete Generativa Avversaria [4] consiste in una coppia di reti neurali artificiali che “competono” tra di loro. Una è un modello generativo G e l'altra un

modello discriminativo D . Queste due reti si comportano similmente ad un contraffattore e un detective: lo scopo del primo è quello di produrre nuovi dati, mentre il secondo apprende come distinguere i dati reali da quelli generati artificialmente (*Figura 5*).

In termini più formali:

- il generatore è una funzione $G : x \rightarrow d_g$, dove x è l'input (può essere un vettore latente, un'immagine o un altro tipo di dato a seconda della GAN utilizzata) e d_g il dato generato;
- il discriminatore è una funzione $D : x \rightarrow p$, dove p è la probabilità che x provenga dalla distribuzione dei dati di addestramento.

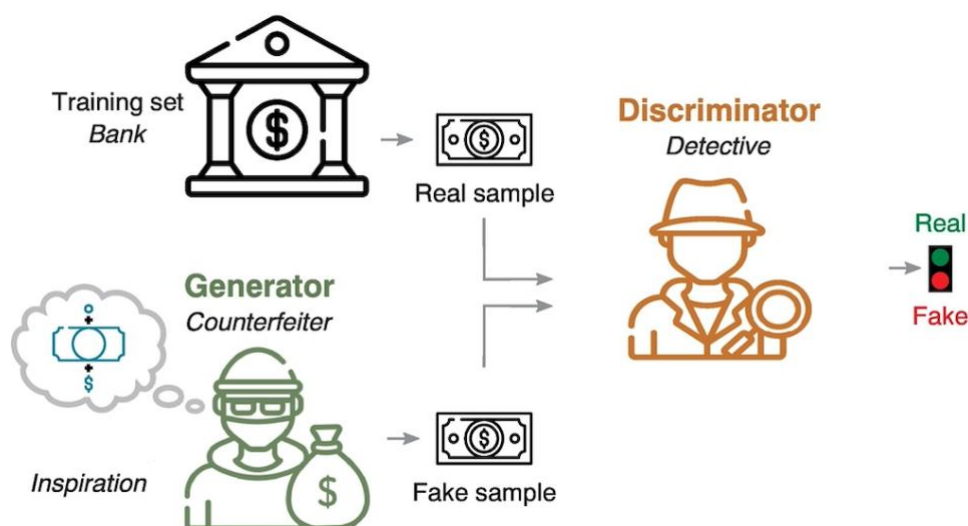


Figura 5: architettura di una GAN.

Lo scopo è quello di ottenere un generatore che sia un buon stimatore dei dati di addestramento. Quando questo avviene, il discriminatore viene "ingannato" e non riesce più a distinguere i campioni provenienti dal generatore da quelli provenienti dal training set.

L'addestramento è effettuato in maniera competitiva: D cerca di massimizzare la probabilità di riconoscere correttamente dati veri da falsi ($\log D(x)$) e G cerca di minimizzare la probabilità che D capisca che ciò che è stato generato non provenga dal dataset di addestramento ($\log(1-D(G(z)))$).

Le due reti vengono addestrate in maniera alternata tramite *retropropagazione dell'errore*, mantenendo invariati i parametri del modello generativo durante l'addestramento del discriminatore e, viceversa, mantenendo invariati i parametri della rete discriminativa durante l'addestramento del generatore.

Le DC-GAN (Deep Convolutional GAN) sono reti generative avversarie che trattano dati nella forma di immagini. Sono usate per generare nuove immagini partendo da una distribuzione di immagini di addestramento.

2.3 CycleGAN

Con le cycleGAN [5][6] si introduce la tematica dell'*Image-to-Image Translation*.

Addestrare un modello per questo compito solitamente richiede dataset popolosi di immagini già accoppiate. Questi dati possono essere difficili e costosi da preparare.

Una cycleGAN, dall'inglese *CYCLE-consistent Generative Adversarial Network*, è un'evoluzione della rete generativa avversaria in cui l'addestramento avviene in maniera non supervisionata. In questo modo è possibile apprendere un modello capace di tradurre un'immagine da un dominio X ad un altro Y, e viceversa, senza dover utilizzare immagini target durante la fase di addestramento, quindi, senza che le immagini dei due dataset siano in relazione tra di loro.

L'architettura della cycleGAN (*Figura 6*) è composta da:

- due generatori $G : X \rightarrow Y$ e $F : Y \rightarrow X$ per tradurre, rispettivamente, immagini dal dominio X al dominio Y e viceversa;
- due discriminatori D_x e D_y , per distinguere gli esempi generati e quelli reali per ciascun dominio.

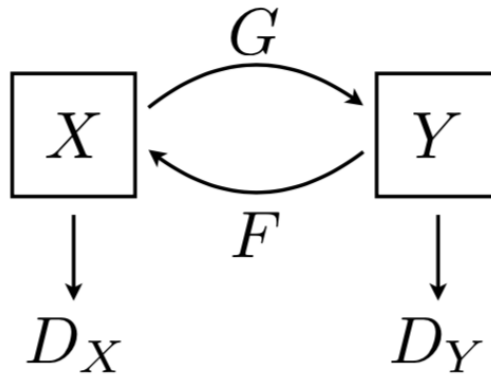


Figura 6: architettura di una CycleGAN.

Una cycleGAN si può immaginare come due reti distinte, una che genera immagini in un senso e l'altra in senso opposto. I generatori e discriminatori sono addestrati in maniera competitiva come le normali GAN.

Queste reti introducono un nuovo tipo di loss per poter aggiornare i pesi delle connessioni: la *cycle-consistency loss*. Quest'ultima compara un'immagine di input alla corrispettiva generata dopo un intero ciclo e calcola la differenza tra le due. Questo processo può essere fatto da X a Y (*forward*) o da Y a X (*backward*). Si elencano i vari passaggi mostrati in Figura 7 con cui si calcola la cycle-consistency loss:

- si passa un'immagine in input x proveniente dal dataset X al generatore G
- x viene modificata da G per ottenere un'immagine che assomiglia a una proveniente dal dataset Y , si ottiene quindi \hat{y}
- si passa l'immagine appena generata \hat{y} al generatore F
- \hat{y} viene modificata per ottenere un'immagine che assomiglia a una proveniente dal dataset X , si ottiene quindi \hat{x}
- la *forward cycle-consistency loss* è la differenza tra l'immagine di partenza x e l'immagine ottenuta dopo un intero ciclo \hat{x} .

Analogamente si calcola la *backward cycle-consistency loss*.

Formalmente i passaggi possono essere espressi in questo modo:

- *forward cycle-consistency*: $x \rightarrow G(x) \rightarrow F(G(x)) \approx x$
- *backward cycle-consistency*: $y \rightarrow F(y) \rightarrow G(F(y)) \approx y$

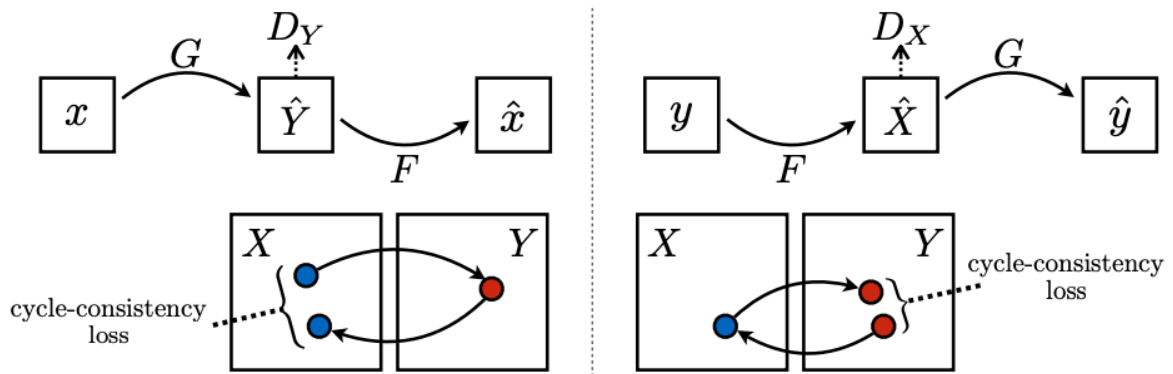


Figura 7: a sinistra forward cycle consistency loss, a destra backward cycle consistency loss.

Quindi, la funzione obiettivo totale sarà data dalla loss della prima rete GAN (quella che va da source a target), sommata a quella della seconda rete (quella che va da target a source) e sommata alle cycle-consistency loss.

Di seguito vengono mostrati alcuni esempi delle applicazioni delle cycleGAN (Figura 8).



Figura 8: esempio risultati cycleGAN.

2.4 StarGAN

L'obiettivo dei modelli per la traduzione di immagine (come le cycleGAN) è cambiare una particolare caratteristica di un'immagine in input con un'altra. Partendo da due dataset d'addestramento con domini diversi, questi modelli imparano a tradurre immagini da un dominio ad un altro.

Un attributo è una caratteristica significativa inerente ad un'immagine come, per esempio, il colore dei capelli, l'età, il sesso etc. Un dominio è un insieme d'immagini che condividono lo stesso valore di un attributo. Per esempio, immagini di donne rappresentano un dominio e quelli di uomini un altro.

Se si volesse costruire un modello per modificare contemporaneamente più attributi di un'immagine, con i metodi visti finora, bisognerebbe costruire, come in *Figura 9*, una coppia di GAN per ogni coppia di domini.

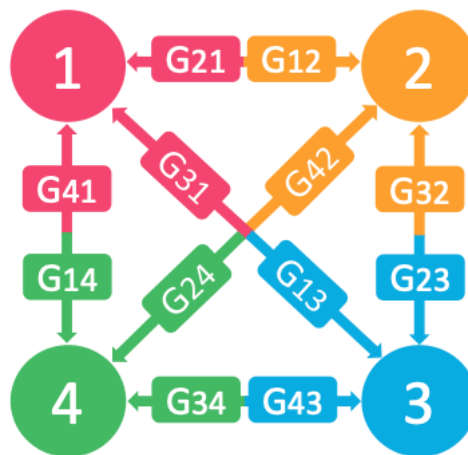


Figura 8: un modello che gestisce molteplici domini utilizzando due GAN per ogni coppia di domini.

StarGAN [7] è modello di *Multi-Domain Image-to-Image Translation* capace di imparare la mappatura fra molteplici domini utilizzando un solo generatore (*Figura 10*). Invece di imparare come effettuare una traduzione di un solo attributo da immagine a immagine, il generatore prende in input sia l'immagine che le informazioni sul dominio e impara come applicare le caratteristiche di quel dominio a quell'immagine.

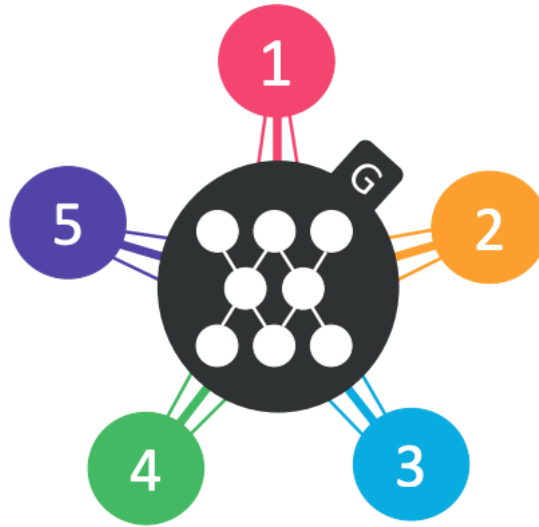


Figura 9: una topologia di starGAN che connette 5 diversi domini.

Il generatore è quindi una funzione $G : (x, c) \rightarrow y$, dove x è l'immagine di input e c è l'etichetta del dominio di target.

Il discriminatore delle starGAN ha due obiettivi:

- stabilire se un'immagine sia vera o artificiale
- predire il dominio di un'immagine attraverso un classificatore ausiliare.

Quindi il discriminatore è una funzione $D : x \rightarrow (p, d)$, dove p è la probabilità che x provenga dalla distribuzione dei dati di addestramento e d è il dominio dal quale il discriminatore “pensa” provenga x .

Si illustra come avviene il processo di addestramento (Figura 10):

- G prende in input sia l'immagine che il dominio che si vuole raggiungere e genera un'immagine sintetica;
- G cerca anche di ricostruire l'immagine originale partendo da quella appena generata usando l'etichetta del dominio originale;
- D produce la probabilità che l'immagine sia finta e la classificazione al dominio corrispondente. In questo modo G cerca di generare immagini indistinguibili dalle reali e che verranno classificate da D con l'etichetta della classe target.

A fine addestramento G produrrà immagini somiglianti ad una corrispondente classe mirata.

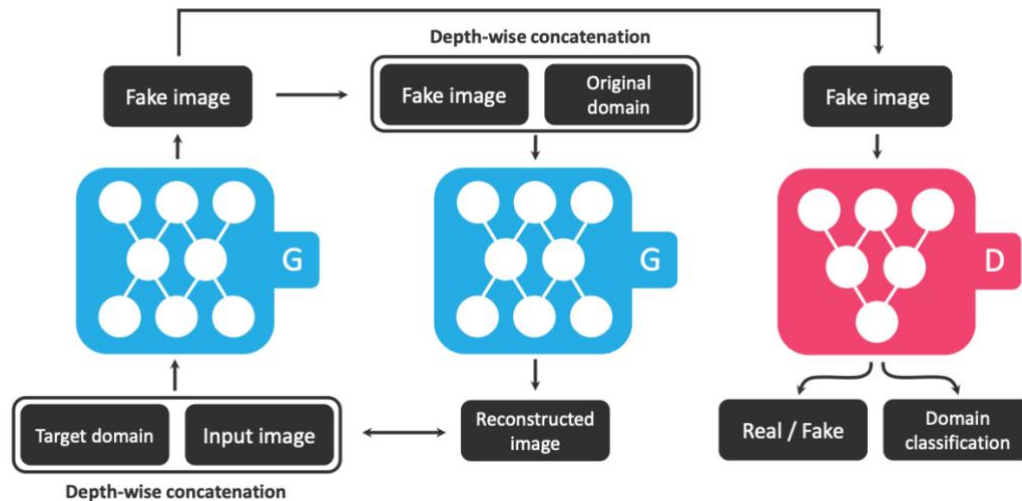


Figura 10: il processo di addestramento di una starGAN.

Con le starGAN quindi si ha a che fare con tre diversi tipi di loss:

- *adversarial loss*: per far sì che le immagini sintetiche generate siano indistinguibili da quelle reali;
- *domain-classification loss*: per far sì che le immagini sintetiche generate appartengano effettivamente alla classe target desiderata;
- *reconstruction loss*: per far sì che le immagini sintetiche generate preservino il contenuto dell'immagine di partenza.

La funzione obiettivo totale è data dalla somma di queste tre.

Un importante vantaggio delle starGAN è che può incorporare simultaneamente molteplici dataset contenenti diversi tipi di etichette.

Per rappresentare il dominio si usano delle etichette. Per esempio, considerando gli attributi uomo, capelli biondi e giovane, la seguente immagine si etichetta con (1, 0, 0).

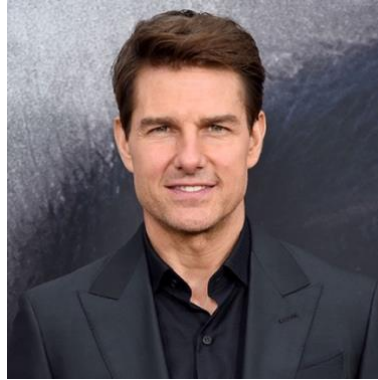


Figura 11: Tom Cruise (1,0,0), immagine presa da Wikipedia/Tom_Cruise.

I dataset utilizzati a loro volta devono essere etichettati con un solo valore impostato ad 1 in modo che in fase di addestramento il modello si possa concentrare sull'attributo desiderato.

Di seguito alcuni esempi di applicazione di starGAN ¹.

Considerando gli attributi (uomo, capelli biondi, giovane):

- nel primo esempio (*Figura 12*) avviene la traduzione da (1,0,0) a (0, 1, 1)
- nel secondo esempio (*Figura 13*) avviene la traduzione da (0,1,0) a (1,1,0).



Figura 12: Tom Cruise - traduzione d'immagine con starGAN.

¹Implementazione starGAN: <https://github.com/sony/nnabla-examples/tree/master/image-translation/stargan>



Figura 13: Jodie Foster – traduzione d'immagine con starGAN.

2.5 Dataset impiegati

2.5.1 Market-1501

Il dataset di immagini Market-1501 è stato raccolto vicino a un supermercato alla Tsinghua University. Sono state utilizzate sei camere e contiene complessivamente 32.668 bounding box relativi a 1.501 individui. Un bounding box è il contenitore con la misura più piccola possibile entro cui sono compresi tutti i punti desiderati, in questo caso la figura di una persona.

Ogni individuo è presente in almeno due camere in modo che possa essere ricercato tra le immagini scattate da diverse camere.

Il pacchetto contiene tre cartelle:

- “bounding_box_train”: contiene 12.936 immagini per l’addestramento;
- “bounding_box_test”: contiene 19,732 immagini per il test;
- “query”: contiene 750 individui re-identificati, ogni individuo compare al massimo in 6 camere per un totale di 3.368 immagini.

Le immagini sono state salvate con l’informazione di camera e di individuo presente nel nome. Per esempio, nell’immagine identificata come “0001_c1s1_001051_00.jpg”, “c1” identifica la prima camera e “0001” significa che si tratta del primo individuo. La *Figura 14* mostra un esempio di immagini proveniente dal dataset Market-1501.



Figura 14: esempio immagini Market-1501.

2.5.2 Duke-MTMC

Duke MTMC (Multi-Target, Multi-Camera) è un dataset basato su filmati di videosorveglianza scattati all'Università di Duke nel 2014. È ampiamente usato per la ricerca e lo sviluppo di sistemi di tracciamento video, re-identificazione di persone e riconoscimento facciale a bassa risoluzione. Il dataset contiene più di 14 ore di video sincronizzato in 8 camere con risoluzione 1080p a 60FPS generando più di 2 milioni di frames dei 2.000 studenti che si spostano nel campus. Il dataset contiene 16,522 immagini di training relative a 702 identità, 2,228 immagini per query relative alle altre 702 identità e 17,661 immagini di test. La suddivisione in cartelle è analoga a quella di Market-1501. Anche in questo dataset le informazioni della camera e dell'individuo sono presenti nel nome dell'immagine.

Numerose pubblicazioni utilizzano il dataset Duke e alcune di queste lo hanno utilizzato per scopi non accademici. Per questo motivo il dataset è oggi difficile da reperire.

In *Figura 15* viene mostrato un esempio di immagini proveniente dal dataset Duke-MTMC.



Figura 15: esempio immagini Duke-MTMC.

2.5.2 MSMT17

MSMT17 (Multi-Scene Multi-Time) è una raccolta d'immagini per la re-identificazione di una persona che rappresenta scenari reali.

Contiene 15 camere (12 outdoor, 3 indoor) che hanno catturato 180 ore di video in 12 slot temporali. Le riprese sono state effettuate nel corso di 4 giorni comprendendo diverse condizioni meteo. Questo implica che le immagini presentano scenari, sfondi più complessi e variazioni d'illuminazione dovute all'ora del giorno (mattina, pomeriggio o notte) in cui sono state scattate.

Il dataset risulta molto più popoloso rispetto agli altri e contiene complessivamente 126.441 bounding box e 4.101 entità. È stato utilizzato come dataset di validazione del sistema di domain generalization proposto.

2.6 Librerie usate

Finora si è parlato di deep learning in maniera teorica; si illustrano ora le risorse utilizzate per mettere in pratica queste tecniche e poter addestrare una rete neurale.

2.6.1 TensorFlow

TensorFlow [8] è una libreria software open-source per l'intelligenza artificiale sviluppata da Google. Può essere usata per diversi compiti ma si concentra particolarmente sull'addestramento di reti neurali per il deep learning.

TensorFlow fornisce un'infrastruttura a livelli per la programmazione differenziabile. La programmazione differenziabile è un paradigma in cui un problema numerico può essere risolto tramite differenziazione automatica. Ciò consente di ottimizzare i parametri di un programma con il metodo della discesa del gradiente. Questo tipo di programmazione è ampiamente utilizzata nel campo delle reti neurali per poter aggiornare opportunamente i pesi delle connessioni. Con *TensorFlow* si può automaticamente ottenere il gradiente di una qualsiasi espressione differenziabile invocando la funzione *GradientTape* per facilitare questo tipo di programmazione.

Praticamente *TensorFlow* è un framework per la manipolazione di array n-dimensionali chiamati *tensors*, variabili e gradienti. Questo framework può godere di accelerazioni GPUs e TPUs.

Mentre *TensorFlow* è una libreria software, *Keras* è un'interfaccia utente per il deep learning che si occupa di strati, modelli, ottimizzatori, funzioni di costo, metriche ed altro. *Keras* è modulare ed estendibile, è disegnata per permettere pratiche e veloci sperimentazioni con le reti neurali rimanendo user-friendly. *Keras* rende le librerie *TensorFlow* produttive.

La classe *Keras Layer* è fondamentale per la creazione di modelli in quanto con le sue sottoclassi incapsula gli stati dei nodi, i pesi e le diverse computazioni che avvengono tra gli strati della rete neurale. Usando i gradienti, le funzioni di ottimizzazione e di loss fornite dall'API si può creare un modello addestrabile.

2.6.2 PyTorch

PyTorch [9] è un framework sviluppato da Facebook, noto per la sua semplicità, flessibilità, efficienza e uso della memoria. È basato su Python e serve principalmente per due scopi:

- rimpiazzare *NumPy* per poter usare la potenza delle GPU e altre accelerazioni;
- fornire una libreria per la programmazione differenziabile utile per le reti neurali.

Per definire una rete neurale artificiale in *PyTorch* si crea una classe che eredita dalla classe base *nn.Module*. Nella nuova classe si definiscono gli strati della rete nella funzione `__init__` e si specifica come i dati passeranno attraverso il network nella funzione *forward*.

Anche in questo framework si lavora manipolando i *tensor*, strutture dati specializzate simili ad array e matrici. Un modello prende in input un dato che viene codificato come *tensor* e ne produce un altro per output.

Torch.optim è un modulo che implementa diversi algoritmi di ottimizzazione usati per costruire reti neurali. Quasi tutti i metodi utilizzati per le ottimizzazioni

sono compresi nelle librerie, quindi, non c'è bisogno di implementarne uno da zero.

2.7 Google Colaboratory (Colab)

Colab è uno strumento per la ricerca e per la didattica basato sul machine learning. Permette di scrivere ed eseguire programmi in linguaggio *Python* nel browser senza il bisogno di alcuna configurazione e con l'accesso gratuito ad accelerazioni GPU.

CAPITOLO 3: Valutazione sperimentale

Prima di passare alle implementazioni vengono definiti alcuni termini necessari per comprendere le impostazioni di un addestramento.

Epoca: è un parametro che indica quante volte i dati dell'intero dataset sono stati passati all'algoritmo di apprendimento. Un'epoca è divisa in più batch. Un modello si dice addestrato quando è trascorso un numero predefinito di epoche.

Dimensione del batch: è il numero di campioni processati prima che il modello sia aggiornato. Alla fine del batch le predizioni sono messe a confronto con le variabili che ci si aspettavano in output e l'errore viene calcolato. A seconda del problema affrontato la dimensione del batch può variare da 1 a dimensione dell'intero dataset.

Normalizzazione: è una tecnica per far sì che le varie caratteristiche siano proporzionalmente significative per la performance dell'addestramento. Senza normalizzazione alcune caratteristiche importanti potrebbero essere ignorate. Si applica ad ogni livello della rete e accelera il processo di addestramento. Ci sono principalmente due tipi di normalizzazione:

- *instance normalization*: opera su un singolo campione, produce risultati ad alta risoluzione ed è utilizzata quando la dimensione del batch è pari a 1;
- *batch normalization*: opera su campioni di dimensione maggiore rispetto alla instance (solitamente ≥ 8) rendendo l'addestramento più veloce anche se si perde in qualità nell'immagine generata.

Per testare i modelli di re-identificazione, sono state usate le seguenti **metriche valutative**:

- mAP (mean Average Precision): $\frac{\sum_{q=1}^Q \text{avgP}(q)}{Q}$, dove Q è il numero delle query presenti nel dataset, e $\text{avgP}(q)$ è la precisione media data una query q ;

- Rank-n: è la percentuale di quante volte il risultato corretto è presente nelle prime n predizioni. Solitamente si considerano per n i valori 1, 5, 10, 20 per n.

Nella Sezione 3.1 e nella Sezione 3.2 sono descritte le due implementazioni che sono state fatte per affrontare il problema della re-identificazione attraverso la generazione di immagini sintetiche.

3.1 CycleGAN per il trasferimento di stile da un dataset ad un altro

Le prime implementazioni ed esperimenti riguardanti le reti neurali sono state eseguite su *Google Colab*.

L'obiettivo della prima implementazione GAN è stato quello di trasferire le caratteristiche di un dataset ad immagini di un diverso dataset. I dataset utilizzati sono quelli visti in precedenza. Sono state considerate circa 10.000 immagini d'addestramento (l'intero training set) per dataset.

In particolare, è stato implementato un modello di *Image-to-Image Translation* che mappa le immagini da Market a Duke e viceversa. Si sono ottenute così 10.000 nuove immagini sintetiche, 5.000 nella configurazione Market → Duke e 5.000 in quella Duke → Market, per la *data augmentation*, prodotte rispettivamente dai generatori *G* e *F*.

Le impostazioni usate per gli addestramenti sono state le seguenti:

- durata addestramento in epoche: 40;
- dimensione immagini: 64x128 (resized quando necessario);
- dimensione massima dataset: 2.000;
- dimensione di batch: 8/16;
- tipo di normalizzazione: batch normalization.

La scelta della *batch normalization* (con numero del batch pari a 8 o 16) è stata dettata dal bisogno di velocizzare il processo di addestramento per via delle limitazioni di *Colab*. Si è deciso, inoltre, di considerare un subset di 2.000

elementi perché, in base a risultati empirici, è stato ritenuto un numero sufficiente per avere risultati in linea con quelli prodotti dall'intero dataset.

Le immagini del dataset Market erano già tutte della dimensione desiderata (64x128), per quelle Duke è stato necessario un resizing.

Vengono mostrati alcuni esempi di risultati ottenuti sia per la traduzione Market → Duke (*Figura 16*) che per quelle Duke → Market (*Figura 17*) comparando le immagini originali con quelle generate dalla cycleGAN.

Market → Duke

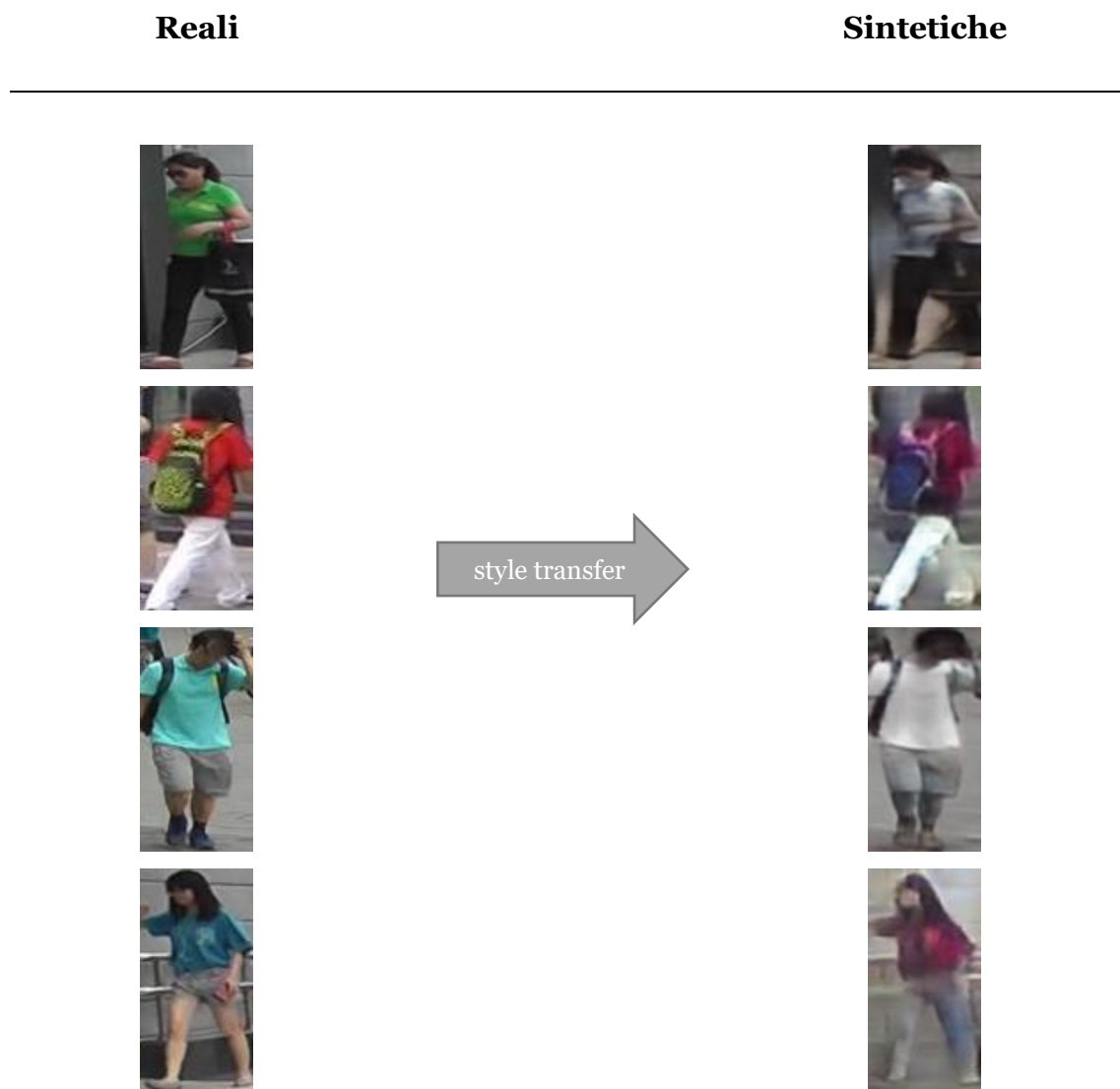


Figura 16: comparazione tra immagini originali e sintetiche generate dalla cycleGAN per Market → Duke.

Duke → Market

Reali

Sintetiche

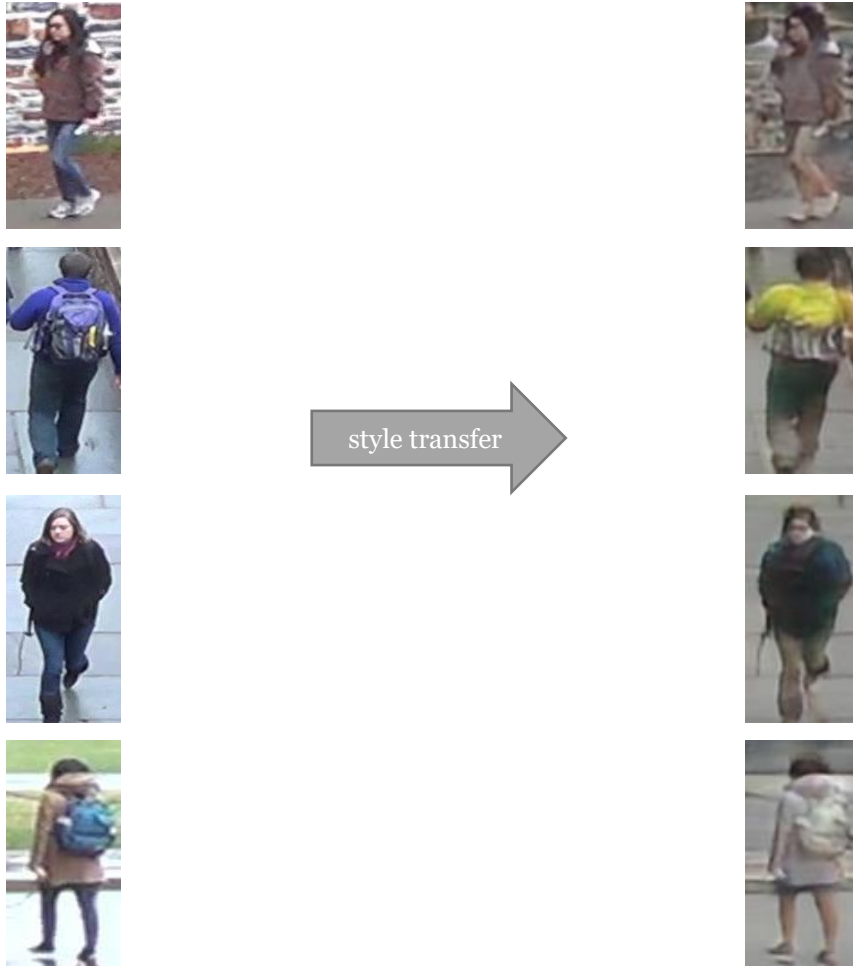


Figura 17: comparazione tra immagini originali e sintetiche generate dalla cycleGAN per Duke → Market.

3.1.1 Discussione risultati

Il modello ottenuto è in grado di generare immagini sintetiche che sembrano reali e mappare le immagini da un dominio all'altro.

Tuttavia, le immagini generate, come si evince dalla *Figura 16* e dalla *Figura 17*, mostrano variazioni troppo evidenti rispetto alle originali perdendo alcune

caratteristiche dell'immagine di partenza. Spesso, infatti, le immagini presentano cambiamenti riguardo il colore degli indumenti e modifiche nella tipologia dei capi indossati. Per cui il riconoscimento del soggetto potrebbe risultare più difficile perché non si può far riferimento all'abbigliamento portato.

Si può ipotizzare che tali variazioni siano dovute al fatto che i due dataset sono stati raccolti in due punti differenti del globo e verosimilmente in due diverse stagioni.

3.1.2 Re-identificazione con data augmentation

Si utilizzano le immagini generate per arricchire i training set dei dataset di partenza.

Avendo già a disposizione un'architettura per la re-ID non resta che addestrarla con i training set ottenuti con la data augmentation, aggiungendo agli di addestramento originali (Market e Duke) le rispettive immagini generate con le GAN per trasferimento di stile da un dataset ad un'altro. I set di training ottenuti sono tutti e due delle dimensioni di circa 15.000 immagini (10.000 originali + 5.000 sintetiche).

L'architettura re-ID che si ha a disposizione è quella implementata da Zhong Zhong per CamStyle [1], in cui viene impiegata come backbone una *ResNet-50*, cioè una rete neurale convoluzionale a 50 livelli di profondità.

Il test consiste nel prelevare un'immagine dal sottoinsieme di query (2.500 per entrambi i dataset) e ricercare l'individuo presente in quell'immagine nelle immagini di test (o di galleria) che comprendono circa 15.000 elementi sia per Market che per Duke.

Il modello è stato prima addestrato con il training set originale e poi con il training set arricchito dalle immagini generate sia per Market che per Duke.

La *Tabella 1* (Market) e la *Tabella 2* (Duke) mettono in comparazione i risultati ottenuti per i due diversi tipi di addestramento.

Market

	dataset originale	dataset originale + data augmentation
mAP	66.1%	55.9%
rank-1	84.8%	89.8%
rank-5	93.9%	93.5%
rank-10	96.1%	93.8%
rank-20	97.6%	96%

Tabella 1: risultati ottenuti addestrando e testando su Market-1501.

Duke

	dataset originale	dataset originale + data augmentation
mAP	48.4%	30.8%
rank-1	65.3%	52.7%
rank-5	79%	68.5%
rank-10	83.9%	73.9%
rank-20	87.7%	78.5%

Tabella 2: risultati ottenuti addestrando e testando su Duke-MTMC.

Dai risultati ottenuti si nota un calo di prestazioni del modello addestrato con i dataset arricchiti rispetto a quello addestrato con quelli originali. Per quanto detto in precedenza questo può essere imputabile al fatto che le immagini prodotte sono troppo eterogenee rispetto alle originali.

3.2 CamStyle

Uno dei principali problemi della re-identificazione di una persona è dato dalle variazioni di camera che, solitamente, si differenziano tra loro per saturazione, risoluzione, illuminazione, etc. Per affrontare questa problematica, in questo lavoro, si è investigata l'architettura CamStyle [1]. Essa è una cycleGAN che cerca di produrre nuove immagini che risultano meno sensibili alle disparità dovute alle diverse camere. In questa maniera, il nuovo training set è una combinazione di tutte le immagini generate dalla cycleGAN per ogni trasferimento di camera.

Nello specifico, CamStyle riconosce a quale camera appartiene un'immagine tramite il nome del file e l'associa ad un dominio. Durante l'addestramento riconosce quali sono le caratteristiche che accomunano tutte le immagini di una

camera per poi trasferirle in un'immagine di un'altra camera. Si tratta di una traduzione di dominio dove i domini sono le varie camere; a tal proposito, in *Figura 18* sono mostrati degli esempi di immagini prese da due camere differenti.



Figura 18 a sinistra un esempio di immagini prese da Market-1501 corrispondenti alla camera 1; a destra dalla camera 6 [1].

Si è implementato CamStyle utilizzando come base il codice scritto da Zhun Zhong ² in *Python* che utilizza il framework PyTorch. È stato riadattato facendo in modo che la traduzione avvenisse da immagini di una camera di un dataset a immagini di un'altra camera di un diverso dataset. Per essere coerente con Market si è deciso di considerare solo le prime sei camere di Duke.

Le impostazioni usate per i diversi addestramenti sono state le seguenti:

- durata addestramento in epoche: 50/32;
- dimensione immagini: 256x256 (resized sempre);
- dimensione massima dataset: intero dataset;
- dimensione di batch: 1;
- tipo di normalizzazione: instance normalization.

I primi addestramenti sono durati 50 epoche ma, considerando che le loss sia dei generatori che dei discriminatori assumevano valori accettabili ai fini della sperimentazione ($< 0,5$) anche in minor tempo, per quelli successivi il numero massimo di epoche è stato impostato a 32. Potendo eseguire il codice su una

² Implementazione CamStyle di Zhun Zhong <https://github.com/zhunzhong07/CamStyle>

macchina con accelerazione GPU è stato deciso di considerare l'intero dataset e optare per la instance normalization (con dimensione di batch 1) per ottenere risultati visivamente migliori.

Si mostrano, in *Figura 19* e in *Figura 20*, alcuni esempi di immagini reali in comparazione a quelle generate da CamStyle.

Market → Duke









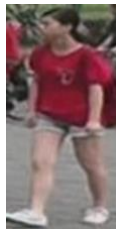









Reali	Sintetiche					
camera 2	camera 1	camera 3	camera 4	camera 5	camera 6	
						
camera 3	camera 1	camera 2	camera 4	camera 5	camera 6	
						
camera 5	camera 1	camera 2	camera 3	camera 4	camera 6	
						

Figura 19: comparazione per camera tra immagini originali e sintetiche generate per Market → Duke da CamStyle.

Duke → Market



Figura 20: comparazione per camera tra immagini originali e sintetiche generate per Duke → Market da CamStyle.

3.2.1 Discussione risultati

I modelli ottenuti sono in grado di generare immagini sintetiche che sembrano reali e trasferire le caratteristiche di una camera ad un'immagine di un'altra camera. Un'immagine catturata da una certa camera viene tradotta con lo stile delle altre. Immagini (reali e non) di una stessa camera presentano le stesse caratteristiche.

Le immagini generate differiscono dall'originale per risoluzione, esposizione, saturazione e luminosità ma mantengono comunque le caratteristiche dell'originale.

Raramente le immagini presentano del rumore, cioè una variazione casuale non desiderata dei pixel.

3.2.2 Re-identificazione con data augmentation

Una volta ottenute le immagini generate da CamStyle il passo successivo è stato quello di usarle per arricchire i training set originali. Sono stati poi utilizzati i nuovi insiemi per addestrare l'architettura re-ID (esposta in precedenza) e infine si sono testati i modelli ottenuti con i corrispettivi set dei due dataset presi in considerazione (Market e Duke).

I set d'addestramento, sia per Market e per Duke, contengono le immagini originali (circa 10.000) + 6.000 immagini sintetiche che comprendono ogni variazione di camera. I set di test e di query sono uguali a quelli utilizzati per gli esperimenti precedenti.

I risultati ottenuti sono esposti nella *Tabella 3* e nella *Tabella 4*.

Market

	training set originale	training set originale + data augmentation
mAP	66,1%	68,5%
rank-1	84,8%	86,4%
rank-5	93,9%	95,1%
rank-10	96,1%	97,7%
rank-20	97,6%	98%

Tabella 3: misure valutative del test sul modello addestrato sul set originale a confronto con quelle del modello addestrato sul set originale arricchito dalle immagini generate Market → Duke con CamStyle.

Duke

	training set originale	training set originale + data augmentation
mAP	48,4%	51,1%
rank-1	65,3%	66,5%
rank-5	79%	81,2%
rank-10	83,9%	86%
rank-20	87,7%	88,7%

Tabella 4: misure valutative del test sul modello addestrato sul set originale a confronto con quelle del modello addestrato sul set originale arricchito dalle immagini generate Duke → Market con CamStyle.

Dai risultati si può affermare che il modello performa meglio, in media di qualche punto percentuale, quando è addestrato con i dataset arricchiti. Possiamo ipotizzare quindi che il metodo utilizzato renda la re-ID meno sensibile alle variazioni di camera.

3.2.3 Re-identificazione cross-dataset

Lo scopo della re-identificazione cross-dataset è poter riconoscere una persona anche quando si ha a che fare con un dataset diverso da quello che si è usato per addestrare l'architettura re-ID.

Per avere una base di comparazione cross-dataset è stata addestrata l'architettura re-ID CamStyle implementata da Zhun Zhong con il sottoinsieme di training di Market e poi testato il modello ottenuto, prima con il corrispettivo test set, poi con le immagini di test dell'altro dataset Duke (*Figura 21*).

Infine, è stato utilizzato il dataset MSMT come dataset esterno di validazione dei test sperimentali.

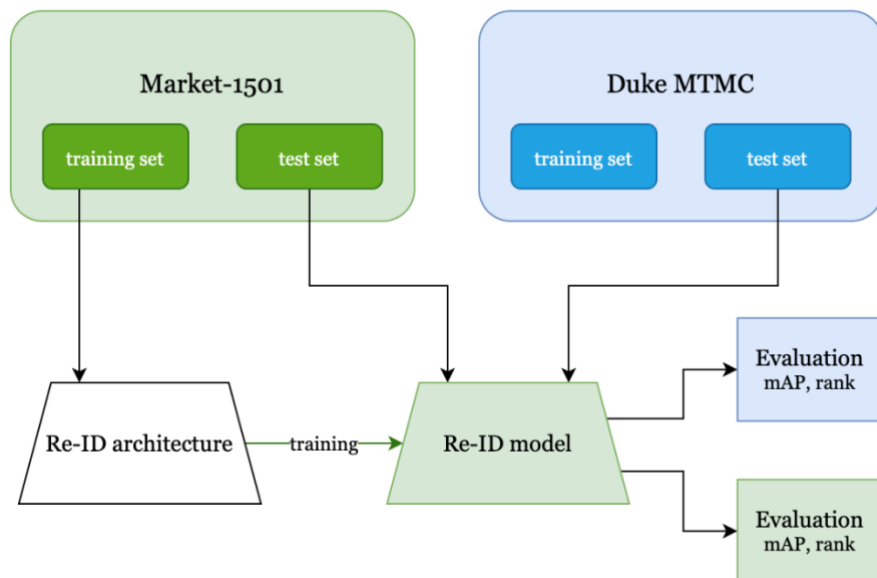


Figura 21: rappresentazione grafica del meccanismo di funzionamento degli esperimenti cross-dataset Market-Duke

In fase di test, il modello ha fornito le performance calcolate con le misure valutative mostrate nella *Tabella 5*.

	training: Market, test: Market	training: Market, test: Duke	training: Market, test: MSMT
mAP	66,1%	18,6%	1,2%
rank-1	84,8%	32%	0,7%
rank-5	93,9%	37,5%	1,8%
rank-10	96,1%	53,8%	3,9%
rank-20	97,6%	61%	8,1%

Tabella 5: risultati dei test effettuati sul modello addestrato con il training set di Market.

Dai risultati si può notare come il modello produca performance soddisfacenti quando viene testato su immagini provenienti dallo stesso dataset (corrispettivo subset di test) con il quale si è svolto l'addestramento, ma cala notevolmente di prestazioni quando viene testato su sottoinsiemi (test) di dataset differenti, come ci si poteva aspettare data l'eterogeneità delle immagini.

Si è effettuata la stessa procedura usando come dataset di training quello di Duke (invece che quello di Market) e in output si sono ottenuti i seguenti risultati (Tabella 6).

	training: Duke, test: Duke	training: Duke, test: Market	training: Duke, test: MSMT
mAP	48,4%	18,4%	1,2%
rank-1	65,3%	41,7%	0,1%
rank-5	79%	59,5%	2,2%
rank-10	83,9%	67,5%	4,0%
rank-20	87,7%	74,9%	8,7%

Tabella 6: risultati dei test effettuati sul modello addestrato con il training set di Duke

Anche in questo caso si notano le scarse prestazioni quando viene testato il modello con un dataset diverso da quello con il quale viene addestrato.

Avendo ora dei dati che esprimono quanto è efficiente il modello addestrato con i set originali, è possibile valutare se le prestazioni migliorano quando si addestra l'architettura re-ID con le immagini sintetiche generate da CamStyle.

Con CamStyle si è addestrata l'architettura CycleGAN in modo tale che si avessero diversi modelli, uno per ogni traduzione di camera da Market a Duke e viceversa. Questi modelli sono stati utilizzati per produrre un nuovo training set di immagini artificiali. In particolare, si considerano per la nuova collezione di dati d'addestramento circa 12.000 (6.000 Market \rightarrow Duke, 6.000 Duke \rightarrow Market) facendo in modo che siano presenti tutte le tipologie di traduzione di camera. Utilizzando il training set di immagini generate è stata poi addestrata l'architettura re-ID per testarla, infine, con i soliti test set e query di Market e Duke. Tutto ciò è schematizzato in *Figura 22*.

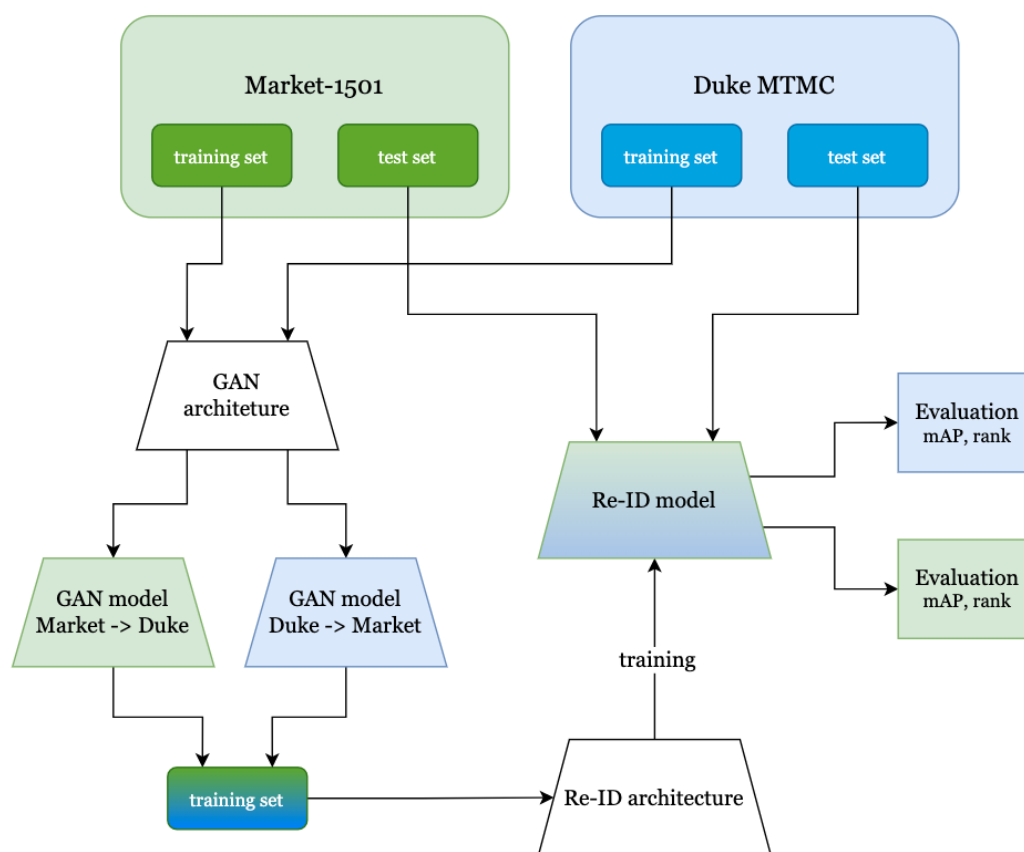


Figura 23: rappresentazione grafica del meccanismo di funzionamento degli esperimenti cross-dataset Market-Duke e della successiva fase di testing del modello re-ID.

Si è deciso inoltre di effettuare un ulteriore esperimento considerando il test set di MSMT17, in modo da verificare un eventuale miglioramento in generalizzazione rispetto al paragrafo precedente. Questo dataset risulta più popoloso avendo circa 80.000 immagini di test e 11.000 query.

La valutazione finale ottenuta addestrando l'architettura con le immagini sintetiche e testandola con i sottoinsiemi dei vari dataset è la seguente (*Tabella 7*).

	test set: Market	test set: Duke	test set: MSMT
mAP	24,1%	21,3%	4,2%
rank-1	49,2%	41,5%	6,4%
rank-5	70,4%	57,9%	10,2%
rank-10	78,4%	64,6%	21,9%
rank-20	85,5%	71,1%	38,1%

Tabella 7: risultati dei vari test sul modello re-ID cross-dataset

Dai risultati ottenuti si può affermare che le prestazioni del modello sui test Market e Duke siano migliorate. Si nota, inoltre, che il test sul dataset MSMT ha prodotto risultati scadenti ma si è verificato, sperimentalmente, che si sarebbero ottenuti risultati peggiori se non si fosse applicato il metodo proposto (si veda la sezione 3.2.2). Queste valutazioni così basse sono probabilmente dovute dal fatto che l'insieme delle immagini raccolte, per le loro caratteristiche, introducono troppa variazione di luminosità e di scenario.

CAPITOLO 4: Conclusioni e sviluppi futuri

Lo sviluppo di questo lavoro ha evidenziato quanto sia ormai di fondamentale importanza l'applicazione dell'intelligenza artificiale, in particolare del deep learning, su problematiche concrete.

Il potenziale dei vari tipi di GAN è vasto e il codice prodotto per gli esperimenti precedentemente esposti non è vincolato, seppur con dovuti adattamenti, al solo campo applicativo della re-identificazione.

In questo elaborato è stata esposta la tematica della re-identificazione di un individuo e proposti dei metodi basati sulle generazioni di immagini sintetiche per la risoluzione del problema delle variazioni di stili e di camere. È stato poi valutato quanto le immagini generate dai vari modelli potessero tornare utili al problema grazie all'investigazione di architetture di re-identificazione preesistenti, raffinate con l'utilizzo di tali immagini.

L'adozione di CamStyle, in particolare, ha permesso in un primo momento di migliorare le prestazioni dei modelli addestrati con immagini reali. Successivamente, ha ulteriormente offerto una miglioria grazie all'utilizzo di immagini sintetiche prodotte dalle GAN, offrendo una probabilità maggiore di identificare correttamente l'individuo.

In conclusione, il metodo proposto complessivamente ha facilitato, permesso di migliorare (per tutti e tre i dataset considerati) la re-identificazione di un individuo, generalizzandola su più domini. In particolare, si sono ottenuti ottimi risultati sui dataset Market e Duke. Tuttavia, i risultati ottenuti utilizzando il dataset MSMT17 come test indipendente, lasciano aperte le possibilità per ulteriori investigazioni orientate allo scopo del *domain generalization* a più ampio spettro.

Per quanto riguarda gli sviluppi futuri sono stati individuati i seguenti punti su cui agire per aumentare le potenzialità del sistema:

- riconoscere ed eliminare le immagini con pixel non volutamente alterati (rumore);

- implementare CamStyle utilizzando l'architettura delle starGAN;
- investigare ulteriori architetture GAN;
- utilizzare per l'addestramento dell'architettura re-ID tutte le possibili immagini producibili con i modelli CamStyle;
- utilizzare l'architettura *Mutual Mean-Teaching: Pseudo Label Refinery for Unsupervised Domain Adaptation on Person Re-identification*³ per migliorare le prestazioni della re-ID cross-dataset soprattutto sul dataset MSMT.

³ Implementazione MMT di Yixiao Ge: <https://github.com/yxgeee/MMT>

BIBLIOGRAFIA

- [1] Zhun Zhong, Liang Zheng, Zhedong Zheng, Shaozi Li, Yi Yang. Camera Style Adaptation for Person Re-identification <https://arxiv.org/pdf/1711.10295v2.pdf>

- [2] Intelligenza artificiale: https://it.wikipedia.org/wiki/Intelligenza_artificiale

- [3] Rete neurale artificiale:
https://it.wikipedia.org/wiki/Rete_neurale_artificiale

- [4] GAN: https://en.wikipedia.org/wiki/Generative_adversarial_network,
<https://machinelearningmastery.com/what-are-generative-adversarial-networks-gans/>

- [5] CycleGAN: <https://machinelearningmastery.com/cyclegan-tutorial-with-keras/>

- [6] Jun-Yan Zhu, Taesung Park, Phillip Isola Alexei A. Efros. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks
<https://arxiv.org/pdf/1703.10593v7.pdf>

- [7] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, Jaegul Choo. StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation <https://arxiv.org/pdf/1711.09020v3.pdf>

- [8] Tutorial TensorFlow: <https://www.tensorflow.org/tutorials>

- [9] Tutorial PyTorch: <https://pytorch.org/tutorials/>