ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA

SCUOLA DI INGEGNERIA E ARCHITETTURA

INSEGNAMENTO DI ARCHITETTURE E PROGRAMMAZIONE DI SISTEMI ELETTRONICI

Confronto tra Algoritmi Classici e di Apprendimento Automatico per Background Subtraction

Relatore:

Chiar.mo Prof. Luca Benini

Autore:

Ludovico Mazzi

Correlatori:

Dott. Alessio Burrello

Dott. Tommaso Polonelli

Dott. Davide Brunelli

IV Sessione - Anno Accademico 2020/2021

Indice

Capitolo 1

Introduzione

La tesi ha l'obiettivo di studiare, implementare e confrontare due diversi algoritmi di *Background Subtraction*: il primo denominato *Gaussian Mixture Model*, più classico, basato ossia su tecniche convenzionali di *computer vision*, il secondo che invece deriva direttamente dall'architettura di una rete neurale, conosciuta come U-Net. In questo capitolo viene presentato il problema, il quale concerne l'analisi, rispetto all'algoritmo tradizionale, del suddetto modello basato su reti neurali, con una panoramica sul contesto di lavoro. Successivamente viene fornita una breve revisione dello stato dell'arte, citando i lavori già svolti, su cui si basa la tesi e che ne costituiscono quindi il punto di partenza. Infine, è definito l'approccio adottato per la risoluzione del problema sopra citato. Il termine *computer vision* [1] indica un campo complesso dell'informatica, il quale racchiude un insieme di problemi che possono essere assimilati alla stessa definizione generale: automatizzare i compiti che permettono la raccolta di informazioni che il sistema visivo umano effettua naturalmente. In pratica, questo vuol dire riuscire a programmare le macchine affinché esse estraggano informazioni dalle immagini. In questa dissertazione ci

si concentra in maniera approfondità sul compito di ottenere una segmentazione degli oggetti, all'interno delle immagini. Questo gruppo di algoritmi è utile allo scopo di quella operazione, definita col nome di *Background Subtraction*, oppure indifferentemente *Image Segmentation* [2]. Esistono però numerose altre abilità che appartengono al campo di lavoro della *computer vision*:

- **Image Classification**: è l'esempio più semplice e rappresenta la capacità di una macchina di catalogare immagini, o parti di un'immagine, classificandole in varie categorie di appartenenza [3]. Dalla letteratura che concerne l'*Image Classification* deriva **Visual Relationship Detection** [4]. Essa è utile per decifrare relazioni condivise fra i soggetti delle immagini.

- **Object Detection**: con questo termine [5] ci si riferisce all'abilità di rilevare gli oggetti di un'immagine in dipendenza dalla loro posizione spaziale. Solitamente l'algoritmo riceve in input uno o più frame e restituisce le immagini in cui gli oggetti sono contornati da rettangoli, accompagnati da un'etichetta, la quale identifica la classe di appartenenza [6].

- **Semantic Segmentation**: questo tipo di segmentazione riesce ad identificare oggetti simili nelle immagini, fornendo una classificazione, basata sulla separazione visuale dei pixel. Quindi la differenza principale con il metodo di *Object Detection* è che invece di utilizzare dei rettangoli per evidenziare gli oggetti, questi ultimi vengono separati dagli specifici pixel di contorno.

Il contributo iniziale, arrivato da tecniche classiche in *pipeline* è stato decisivo per lo sviluppo di questo settore dell'informatica. Allo stesso tempo, è essenziale evidenziare i limiti di questo approccio *"tradizionale"*, nei confronti delle nuove sfide accolte.

Gli algoritmi basati su reti neurali si sono rivelati potenti strumenti per effettuare tecniche di *Background Subtraction*, sia su singole immagini, così come su video acquisiti tramite telecamere statiche. In questo contesto si colloca lo studio della tesi su più tipologie di reti neurali, che si conclude con l'implementazione di un algoritmo, appartentente alla classe delle *Convolutional Neural Network*, promettente in termini di precisione. Quest'ultima è misurata basandosi sulla metrica, denominata *pixel difference*, nella quale si effettua una differenza assoluta nei valori dei singoli pixel, che compongono le immagini in uscita di GMM e della rete.

Partendo dalla base dei lavori precedenti, ci si concentra sul problema della scelta fra l'utilizzo di algoritmi classici e reti neurali, evidenziandone punti di forza e debolezze, nel caso di applicazioni per *Background Subtraction*. L'obiettivo è fornire criteri di valutazione per decidere il miglior compromesso fra affidabilità del codice e la relativa semplicità di implementazione. Nel seguente elaborato è stata fornita, in primo luogo, una panoramica generale dello stato dell'arte, rappresentata dall'algoritmo di *Gaussian Mixture Model*. Successivamente viene giustificato l'utilizzo ipotetico di alcuni tipi di algoritmi, quali esempi di reti neurali applicabili nell'ambito della *computer vision*. Si esplorano quindi le specificità di classi di modelli, a cui appartengono reti neurali più o meno "profonde". In alcuni casi, segue un'analisi più approfondita, di tipo anche quantitativo. Si propende così per l'utilizzo di un'architettura appartenente alla classe delle *Convolutional Neural Network*, in particolare direttamente derivata da U-Net. Essa è una rete a cinque livelli, in cui quelli intermedi si occupano di effettuare l'operazione di convoluzione. I dati ricavati dall'analisi, confermano che quest'ultimo modello raggiunge livelli di accuratezza pressoché identici ai risultati ottenuti tramite *Gaussian Mixture*

Model. L'accuratezza viene calcolata tramite la differenza dei valori dei pixel, all'interno delle immagini ottenute come output dei due algoritmi. Quest'ultimi, si discostano fra loro di appena il 2%, nei casi peggiori. La dissertazione rivela che è possibile sostituire il dataset specifico considerato, seppur l'algoritmo più classico ottenga risultati soddisfacenti, con il modello di rete neurale sviluppato per il lavoro di tesi, più generalizzabile. Di fatto quindi, i contributi principali della tesi sono:

- Lo studio preliminare della letteratura, in merito all'utilizzo di algoritmi di reti neurali.

- L'implementazione di un algoritmo di *Frame Difference* per ottenere *Background Subtraction*. Esso si pone l'obiettivo di costituire la base di partenza, al fine di introdurre il *task* specifico.

- L'implementazione di un algoritmo denominato *Gaussian Mixture Model*. Esso è considerato, all'interno della dissertazione, come lo "stato dell'arte" per il compito di *Background Subtraction*. Il *framework* implementato fornisce inoltre le immagini segmentate, le quali saranno input del modello neurale.

- L'implementazione di un modello, che si classifica come *Convolutional Neural Network*. L'architettura è ricavata da U-Net ed è formata da livelli di convoluzione 2D. Esso effettua lo stesso compito di GMM, utilizzando le immagini iniziali e le immagini segmentate come input della rete.

- Infine, l'analisi delle prestazioni dell'algoritmo basato sull'intelligenza artificiale, rispetto a GMM. Il metodo utilizzato per valutare le prestazioni è detto di *pixel difference* e si basa sul calcolo di una differenza assoluta nei valori dei pixel che formano le immagini. Il modello di CNN garantisce una precisione in media, nel restituire i frame segmentati, del 99.3%, se confrontato con i risultati di GMM. Il primo, rappresenta quindi un'implementazione migliorativa per questo specifico compito.

Capitolo 2

Retrospettiva

2.1 Background Subtraction

Negli ultimi trent'anni la tecnica di *Background Subtraction* è stata uno dei temi di ricerca più importanti nell'ambito della *computer vision* per supportare numerose applicazioni come la sorveglianza intelligente, il monitoraggio del traffico e la supervisione delle macchine industriali. Con il termine *Background Subtraction*[7] si intende identificare quel metodo per isolare le parti in movimento di una scena, dividendo quest'ultima in *foreground*, la parte utile e *background*, da eliminare.

Figura 2.1: frame e rispettiva Background Subtraction

Il metodo più semplice consiste in una differenza, in termini assoluti, fra il frame corrente e quello di riferimento per il *background*. Se il valore del pixel considerato supera quello della soglia di riferimento, allora esso appartiene al *foreground*, altrimenti fa parte dello sfondo della scena. In gran parte delle applicazioni, in cui non viene fornito il fotogramma di sfondo, è necessario ricavare un modello di *background*, che si aggiorna costantemente. Quindi il modello di sfondo viene la prima volta inizializzato, successivamente dopo ogni sottrazione, aggiornato. Il funzionamento può essere così schematizzato in pochi passi:

- stima del *background* al tempo t
- differenza tra immagine di *background* e frame corrente
- confronto tramite differenza assoluta con un valore di soglia, il cui risultato costituisce la *maschera* di *foreground*

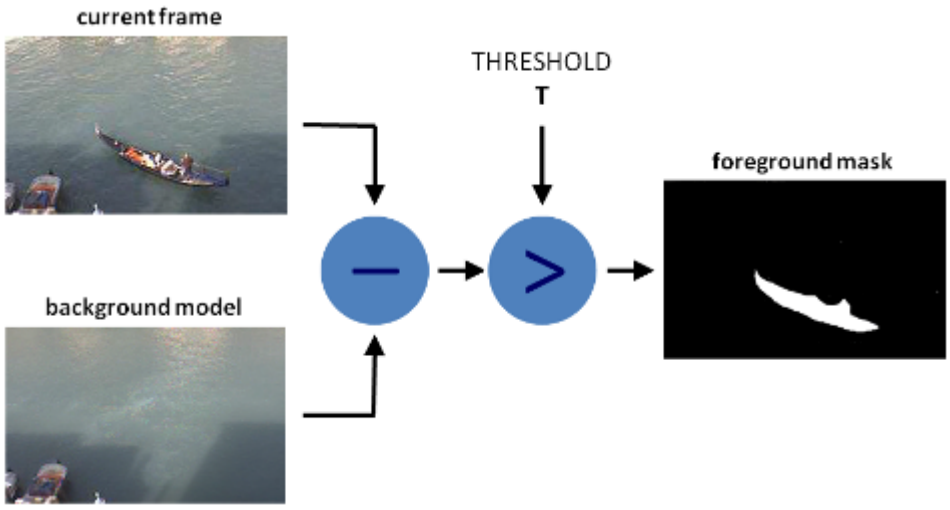


Figura 2.2: Background Subtraction basato su una soglia

2.2 Rete neurale

Ancora prima di descrivere il compito della tesi, è necessario proiettarsi nel contesto di lavoro, introducendo il concetto di *neural network*. Gli algoritmi basati su reti neurali **(NN)** sono una branca dell'intelligenza artificiale. Essi si basano sull'idea che i sistemi automatizzati possano apprendere dai dati, identificare i modelli ottimali autonomamente e prendere decisioni, riducendo l'intervento umano al minimo. Le reti neurali sono modelli parametrici che svolgono una serie di operazioni sequenziali sui dati in ingresso. Sono composte da un insieme di unità fondamentali connesse tra loro, a formare una rete. Queste unità vengono dette ***nodi*** o anche, ispirandosi alle reti neurali in ambito biologico, ***artificial neuron***. Analogamente ogni connessione tra i neuroni è chiamata ***edge*** e rappresenta la sinapsi del cervello. Così come i *nodi* della rete possono ricevere segnali dagli altri a cui sono connessi, allo stesso modo essi possono essere mittenti dei segnali processati. Per segnali si intendono numeri reali calcolati tramite diverse funzioni non lineari. Ciascun *neurone* e ciascuna connessione della rete hanno tipicamente dei pesi (***weight***) che modellano il processo di apprendimento automatico. Sostanzialmente i pesi aumentano o diminuiscono la forza che acquisiscono i segnali attraverso ciascuna connessione. In maniera pratica i *neuroni* sono aggregati tramite livelli (***layer***), per cui ciascuna rete neurale presenta una propria architettura che consiste di differenti livelli, che vengono infatti classificati come *input layer*, ossia livelli di ingresso, *hidden layer*, intendendo con questi tutti i livelli intermedi che intercorrono fra l'ingresso e l'uscita, e infine appunto, l'*output layer*, ossia lo strato a conclusione della rete.

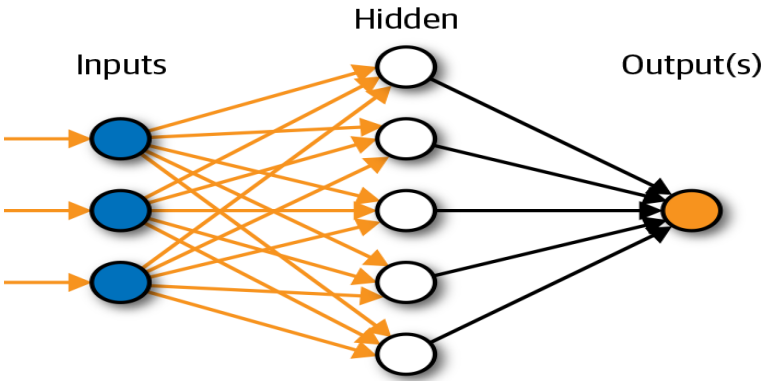


Figura 2.3: Rete neurale "shallow"

Ogni livello consiste in una trasformazione lineare seguita da una funzione di attivazione generalmente non lineare. Le funzioni di attivazione più ampiamente utilizzate sono:

- sigmoide
- ReLU
- PReLU

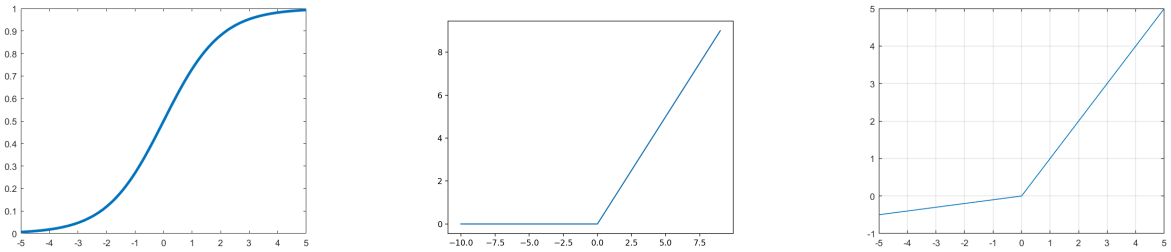


Figura 2.4: Da sinistra verso destra: sigmoide, ReLU, PReLU

Ciascuna funzione di attivazione può essere vista come la mappatura non lineare degli ingressi sulle uscite, tramite ciascun nodo. I parametri fondamentali della

rete sono aggiornati tramite la definizione di una funzione di costo: in pratica si definisce una certa funzione di costo e lo scopo del codice è minimizzare il valore che essa assume per il risultato specifico che si vuole ottenere. Il metodo più comune per il processo di apprendimento è la *back propagation*, quindi viene calcolato il gradiente della funzione d'errore e l'uscita viene di seguito propagata all'indietro per aggiornare i parametri. Insieme alla *back propagation*, altre tecniche, come la *batch normalization*, permettono di accelerare l'apprendimento della rete, incrementandone la velocità e regolarizzandolo. Tutto questo avviene attraverso *epoch* successive. Il termine non va confuso con una singola iterazione, infatti i due concetti sono uguali solo nel caso in cui i parametri vengano aggiornati una volta ogni passaggio attraverso l'intero *dataset*.

Il successo empirico del *deep learning* non sminuisce le numerose sfide affrontate presso i teorici. In particolare i punti critici evidenziati da uno studio del 2018 condotto da Vidal et al. [8] sono tre: l'architettura, le tecniche efficaci di regolarizzazione e i giusti algoritmi di ottimizzazione. Capire la congiuntura migliore di questi aspetti e le loro intercorrelazioni è fondamentale per il successo della rete neurale:

- **Architettura**: il numero, la misura, il tipo di *layer* sono la chiave di un'architettura. Il problema principale è come una data architettura impatta sull'espressività, che è l'abilità di approssimare funzioni arbitrarie dell'input. In questo senso, numerosi studi hanno dimostrato la maggiore efficacia delle *deep neural network* (costituite da più di tre livelli) rispetto alle *shallow neural network* (reti neurali non profonde), senza perdita di generalità nei confronti della cattura di aspetti invarianti dei dati.

- **Ottimizzazione**: questo aspetto coinvolge l'allenamento della rete e contiene in realtà due aspetti: il *dataset* utilizzato e in molti casi anche l'algoritmo utilizzato per ottimizzare la rete. É indubbio che il problema sia non convesso e i problemi maggiori riguardano la garanzia del caso ottimo. Le reti neurali sono modelli non convessi.

- **Proprietà di generalizzazione e regolarizzazione**: la più grande preoccupazione è come sia possibile generalizzare una *deep neural network* e come essa debba essere regolarizzata prevenendo *overfitting*. L'*overfitting* è quel fenomeno che caratterizza reti neurali che sono state addestrate più del necessario ed ottengono ottime prestazioni sul *dataset* utilizzato per l'allenamento, ma pessimi traguardi sui nuovi dati. Così come l'*underfitting*, che è il problema opposto, questo fenomeno peggiora la generalità della rete. La sfida maggiore riguarda l'abilità della rete di generalizzare da un piccolo numero di esempi ottenuti durante la fase di *training*.

- **Proprietà di stabilità**: in aggiunta, l'instabilità dell'output di una rete neurale può essere dovuta a piccole perturbazioni dell'input che possono distorcere in maniera anche significativa i parametri. La stabilità può essere incrementata in un primo momento utlizzando pesi gaussiani randomici, o in maniera maggiore, tramite tecniche di *forward propagation* ispirate dai sistemi di equazioni differenziali ordinarie e da tecniche efficaci di normalizzazione dei pesi.

Di seguito viene fatta una breve rassegna di alcune tipologie di architetture, utilizzate nell'ambito delle rete neurali:

- ***Restricted Boltzmann machine* (RBM)**: è un'architettura [9] relativamente semplice. Si tratta di una rete neurale formata da due livelli, i quali sono connessi completamente a formare una grafico bipartito. Ciò significa che ogni nodo del livello di input è connesso con ciascuno dei nodi dell'*hidden layer*, che costituisce il secondo livello, mentre non sono presenti interconnessioni all'interno dello stesso livello. RBM è una rete neurale stocastica, ciò significa che le funzioni di attivazione della rete, che hanno di solito attivazione binaria, dipendono, in questo caso, da una funzione di densità di probabilità.

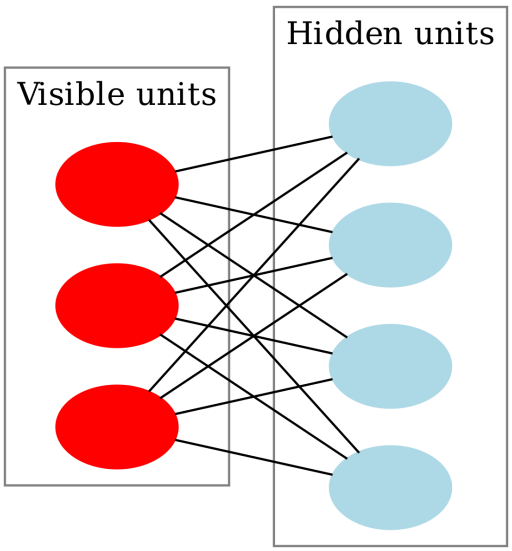


Figura 2.5: Architettura di RBM

- ***Deep Belief Network* (DBN)**: è già un esempio di *deep neural network* [10]. Essa è ottenuta tramite una cascata di RBM a formare una pila. Il pri-

mo passo è far apprendere un insieme di proprietà, al primo livello, ottenute dal segnale in ingresso, direttamente dai pixel. Il passaggio successivo è trattare i valori di questo livello, che sono stati acquisiti, come pixel e ripetere quindi il procedimento per il secondo livello, e così via per i successivi. Ogni volta che viene aggiunto un livello alla rete, si verifica un miglioramento di base delle prestazioni.

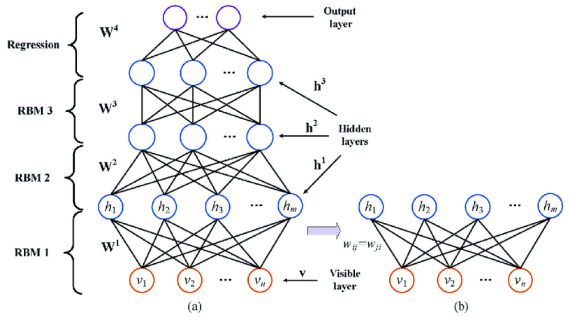


Figura 2.6: Architettura di DBN

- ***AutoEncoders Network* (AE)**: in questa architettura i dati in ingresso sono convertiti tramite rappresentazioni astratte, poi convertite nel formato originale usando una funzione di decodifica. In termini pratici, AE viene allenata per imparare a codificare l'ingresso in una rappresentazione da cui l'input può essere ricostituito. Il vantaggio di questa procedura risiede nel fatto che AE riesce ad estrarre così, continuamente, *feature* utili al processo, filtrando invece le informazioni non necessarie per l'applicazione specifica [11]. Le prestazioni, durante il processo di apprendimento, risultano migliorate in quanto il dato in ingresso è trasformato in una rappresentazione di dimensioni minori. In particolare, AE ha dimostrato la sua efficacia per proprietà non lineari, nel contesto di problemi di varia natura. In ogni caso, è vero che nei problemi reali, spesso i dati sono corrotti da rumore e *outliers*.

Per trovare una soluzione, nel 2017, Zhou e Paffenroth [12] hanno proposto l'uso di AE resistenti ai disturbi, basati su RPCA. In pratica, il dato in input viene diviso in due parti: la prima, che può essere successivamente ricostruita da un AE e la seconda che contiene appunto gli *outliers* e il rumore del dato originale. Questo nuovo modello è stato chiamato *Robust Deep AutoEncoder* (RDA).

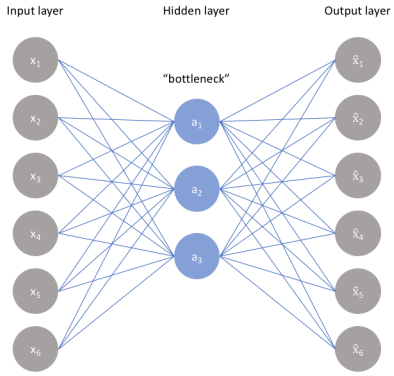


Figura 2.7: Architettura di AE

- ***Deep Convolutional Neural Networks* (CNN)**: sono una sottocategoria delle *discriminative deep architecture* e si sono rivelate efficaci nel processare dati 2D come immagini o video. L'architettura [13] è ispirata alla corteccia visiva degli animali e si basa sul concetto di *time-delay neural network* (TDNN). Infatti le informazioni della stessa natura vengono processate in parallelo, la convoluzione rimpiazza la matrice generale di moltiplicazione. CNN rappresenta una soluzione per ridurre le immagini in una forma in cui sia più facile processarle, senza però perdere informazioni, che sono utili per effettuare delle predizioni affidabili. Questa caratteristica è utile al di là della singola applicazione, ma piuttosto ne deve essere valutata la scalabi-

lità verso *dataset* di ordini di grandezza maggiori. Questa architettura verrà approfondità nei capitoli successivi.

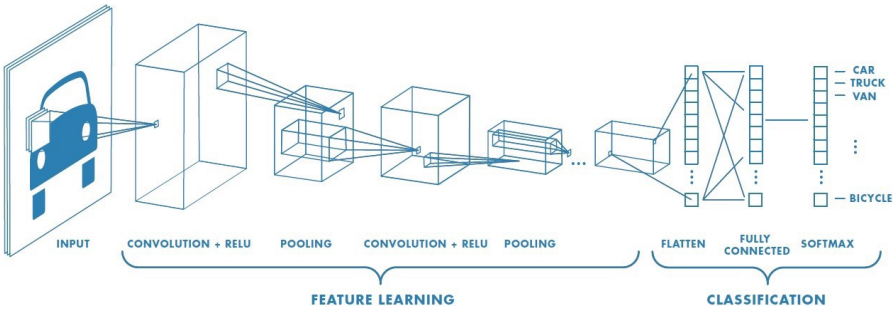


Figura 2.8: Architettura di CNN

- ***Deep Fuzzy Neural Network* (DFNN)**: Basata sul concetto di *fuzzy learning* [14]. In molte applicazioni reali, le risposte ai problemi non possono essere di tipo binario (0 e 1), ma richiedono un certo livello di incertezza. DFNN opera in assenza di confini netti fra le informazioni ed anzi, l'incertezza viene convertita in un gradiente di valori che spaziano da un valore limite all'altro. Così il passaggio fra 0 ed 1 non avviene per scatti, ma attraverso una transizione graduale. Derivando le informazioni sia da rappresentazioni *fuzzy* e neurali. In questo modo, la conoscenza acquisita da entrambe le tipologie è fusa, restituendo una rappresentazione finale da classificare.

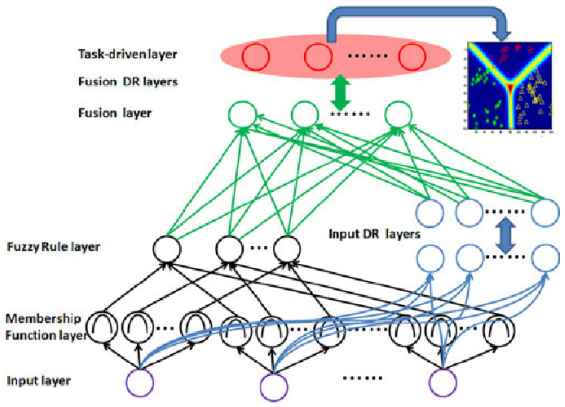


Figura 2.9: Architettura di DFNN

- ***Generative Adversarial Network* (GAN)**: Introdotta nel 2016 [15], essa fornisce un potente strumento per utilizzare dati non classificati, nel *training* di modelli di machine learning ed è diventata uno dei più promettenti paradigmi per l'apprendimento *unsupervised*. Più precisamente, utilizzando un processo avversario, in cui due modelli sono allenati: un modello così detto *generativo*, che cattura la distribuzione dei dati, ed un modello *discriminante*, che stima la probabilità che un campione sia derivato dall'insieme dei dati dell'allenamento, piuttosto che dal modello *generativo*. Diventa possibile così massimizzare la probabilità che il modello *discriminante* commetta un errore.

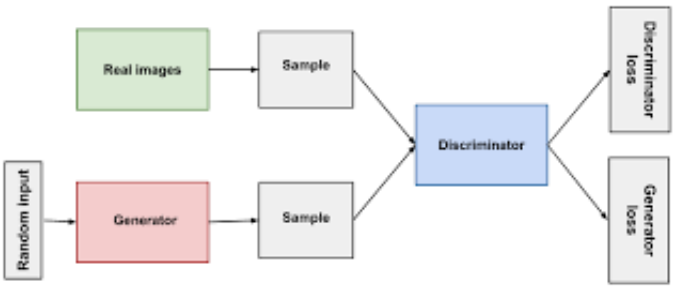
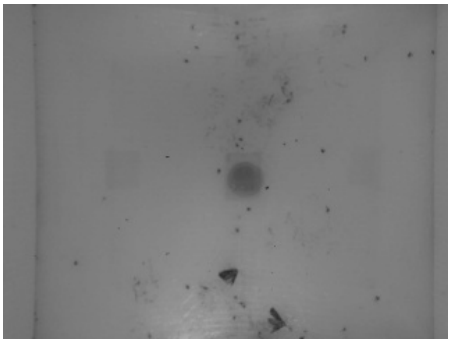
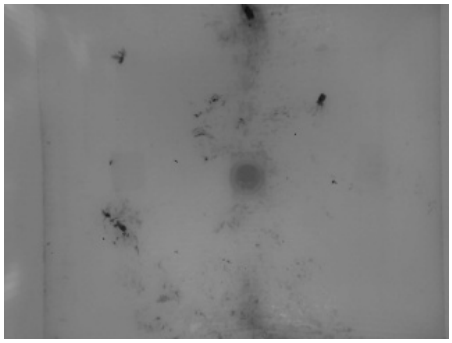
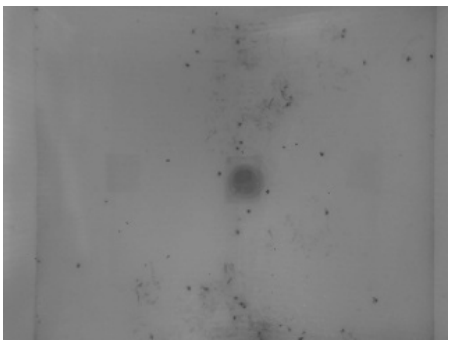
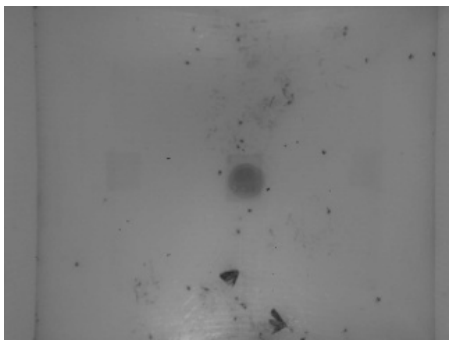
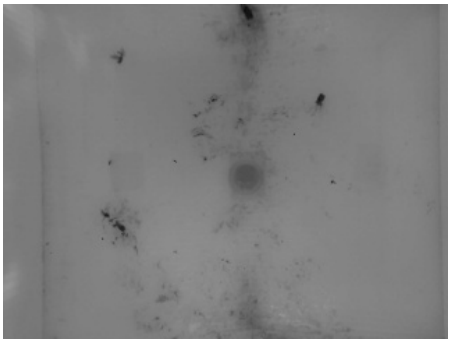
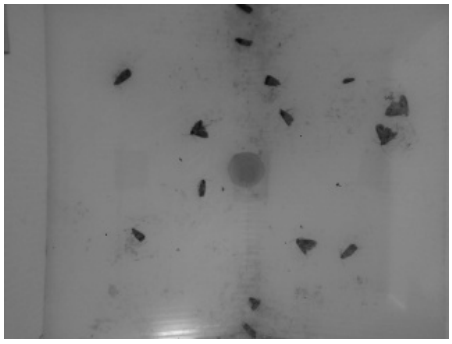
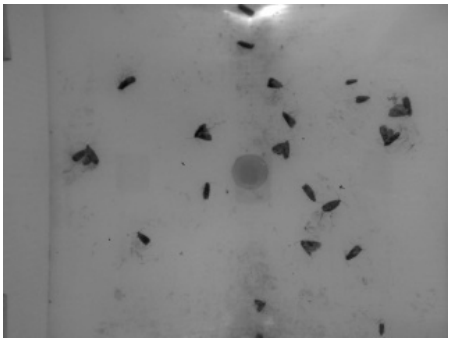
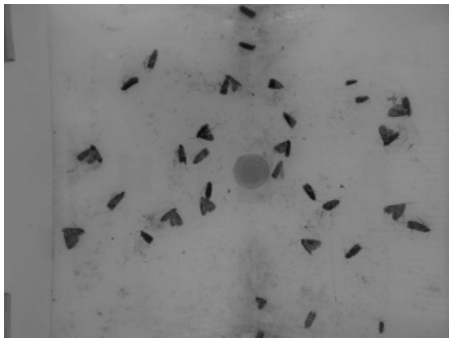


Figura 2.10: Architettura di GAN

2.3 Appendice sul dataset

Il *dataset* utilizzato nel caso di questo lavoro di tesi è stato fornito dai ricercatori della facoltà di Agraria, dell'ateneo di Bologna. Esso rappresenta una sequenza di fotogrammi catturati da una videocamera, puntata su una trappola per insetti. Essa è costituita da un foglio bianco cosparso di colla, usato come sfondo. Le immagini non sono in questo caso binarie, per questo motivo, spesso si è reso utile convertirle in questo senso. La conversione non diminuisce la quantità di informazioni utili e permette di avere un minor peso computazionale. Il *dataset* in questione è composto da 56 differenti fotogrammi. Si nota una tendenza generale, che va verso un aumento progressivo del numero di oggetti (insetti in questo caso) in ogni immagine.

Capitolo 3

Studio delle principali tecniche considerate

In questo capitolo viene fornita una revisione dei principali algoritmi considerati, in una prima fase di ricerca, come candidati per la risoluzione del problema.

3.1 Gaussian Mixture Model

Gaussian Mixture Model **(GMM)** [16] è un algoritmo di *"soft" clustering* che non prevede, perciò, un approccio deterministico alla risoluzione del problema, ma piuttosto probabilistico: si assume, infatti, che siano presenti un certo numero di distribuzioni gaussiane e che ognuna di esse rappresenti un *cluster*. Una distribuzione gaussiana è, appunto, una distribuzione di probabilità continua spesso utilizzata come prima approssimazione per descrivere variabili casuali a valori reali che tendono a concentrarsi attorno ad un singolo valor medio. Il grafico della funzione densità di probabilità associata è simmetrico ed ha una forma caratte-

ristica a *"campana"*. Si raggruppano di conseguenza le osservazioni appartenenti ad una stessa distribuzione, in termini di probabilità, in uno stesso *cluster*. Ogni distribuzione, essendo di tipo gaussiana, è identificata da due parametri statistici:

- il valore medio μ
- la varianza σ

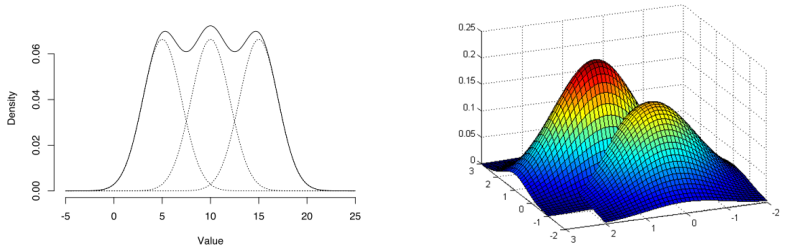


Figura 3.1: grafico in 2D ed in 3D di GMM

è interessante notare che rispetto al *k-means* in cui interviene solo il contributo del valor medio, nel GMM è presente anche la varianza, questo permette l'identificazione di *cluster* anche non di forma sferica, mantenendo però lo stesso approccio e gli stessi vantaggi del *k-means*. L'algoritmo prevede che, una volta definite le distribuzioni presenti, ovvero il numero di cluster della partizione del *dataset*, il processo iterativo sia realizzato grazie ad un algoritmo statistico, detto di *Expectation-Maximization* (EM), il quale si occupa di calcolare e aggiornare i parametri coinvolti.

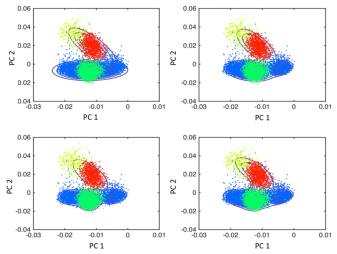


Figura 3.2: clustering di GMM

EM consta di due fasi:

- **fase E**: passaggio in cui si calcola la probabilità di appartenenza, per ogni oggetto del dataset, ad ogni distribuzione, ovvero ad ogni cluster
- **fase M**: passaggio in cui sono aggiornati i parametri propri della distribuzione, sulla base dei risultati ottenuti durante la prima fase; dunque, il valore medio, la funzione densità di probabilità e la covarianza di ogni distribuzione gaussiana saranno pesate sulle probabilità di appartenenza di ogni singolo oggetto.

Tale processo viene ripetuto un numero prefissato di iterazioni al fine di massimizzare la funzione di verosimiglianza di ogni dato contenuto nel *dataset*.

La distribuzione gaussiana viene usata con lo scopo di rendere l'algoritmo più resistente rispetto a variazioni di luce. Le gaussiane multiple vengono analogamente utilizzate per fronteggiare cambiamenti rapidi nella scena, come per esempio il movimento rapido delle foglie sugli alberi. Poiché i singoli punti delle immagini in una scena sono considerati indipendenti tra loro, allora ciascun punto in un dato momento del tempo è modellato tramite GMM. Considerando quindi il caso particolare, il singolo punto è identificato da una variabile randomica x con diver-

si valori nel tempo. La GMM è proprio un insieme di k distribuzioni gaussiane. Definiamo ora una funzione $Pr(x)$:

$$\boldsymbol{Pr(x) = \sum_{k=1}^{K} w_k \times \rho(x; \mu_k, \sigma_k)}$$

dove ovviamente i pesi sommati devono restituire uno e ρ_k rappresenta la $k-esima$ curva gaussiana.
Se ora consideriamo lo stesso punto visto in diversi istanti di tempo allora ogni punto avrà la sua $Pr(x)$. Il processo relativo alla *background subtraction* attraverso l'algoritmo di GMM si effettua in più fasi [17]:

- si inizializza considerando una gaussiana relativa al primo istante di tempo, a cui è associata la variabile x e si assegna di conseguenza peso unitario.

- ora, procedendo tramite metodo iterativo, viene controllato se c'è almeno una gaussiana che genera x_t. In caso positivo vengono aggiornati tutti parametri delle gaussiane, quindi: valor medio, varianza e learning rate. In caso negativo, invece, viene rimossa la gaussiana con meno peso, se sono già presenti k gaussiane, poi si aggiunge la distribuzione creata, alle $k-1$ già presenti.

Il processo di sottrazione consiste nel selezionare la gaussiana che stima il rapporto più grande w/σ. Si seleziona la prima gaussiana come modello di *background* tale che valga l'espressione:

$$\boldsymbol{B = argmin_b(\sum_{k=1}^{b} w_k > T_b)}$$

Quelle maggiori di un certo valore di soglia T_b sono quindi identificate come gaussiane relative al *background*. Dopo aver determinato le gaussiane relative allo sfondo, in quel momento, se rispetto al punto in quell'istante di tempo t c'è almeno una curva gaussiana relativa allo sfondo allora quel punto sarà perciò di *background*, altrimenti il punto assume l'etichetta di *foreground*.

3.2 Convolutional Neural Network

Riprendendo i capitoli precedenti, una ***Convolutional Neural Network*** è un algoritmo di *deep learning* che riceve in ingresso immagini ed è capace di differenziare vari aspetti di esse [18]. La preprocessazione richiesta in una CNN è minore se comparata ad altri algoritmi di classificazione. Questo tipo di rete neurale riesce, attraverso la fase di addestramento, ad imparare le caratteristiche peculiari dell'immagine e dei filtri.
Una CNN ha la specificità di catturare le dipendenze spaziali e temporali in un'immagine, attraverso l'applicazione di alcuni filtri.
Consideriamo per ora una immagine RGB, separata nei suoi tre colori (rosso, verde e blu). Se consideriamo una figura reale, è facile capire quanto il compito sia computazionalmente intensivo. La CNN diminuisce la dimensione delle immagini in una forma più facilmente processabile, senza perdere però proprietà fondamentali per una giusta predizione.

A titolo di esempio, consideriamo ora invece un'immagine binaria (un solo canale colore). L'elemento che viene utilizzato per applicare l'operazione di convoluzione, nella prima parte del livello di convoluzione, è detto *kernel* (filtro), anch'esso è una matrice bidimensionale, generalmente di dimensione minore. Il

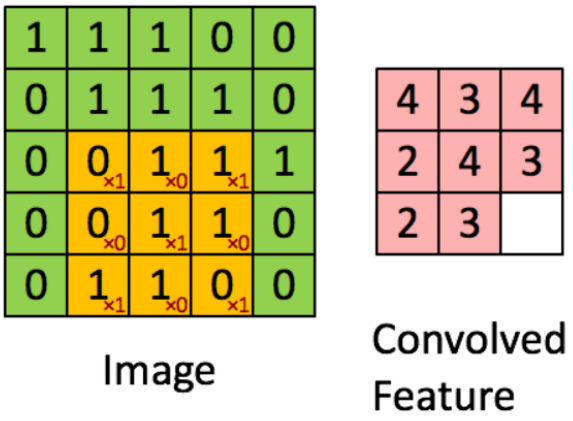


Figura 3.3: Kernel

kernel si sposta su ogni pixel del frame, calcolando una moltiplicazione matriciale fra lo stesso filtro e la porzione di immagine considerata. In particolare il *kernel* si muove seguendo il valore che assume il parametro di *stride*, il quale detta quanti pixel saltare. Questa procedura avviene finché non è stata completata l'intera immagine. L'obiettivo di queste operazioni è estrarre dettagli di così detto "alto livello", come spigoli e gradienti di colore. Con l'aumentare della profondità della rete, aumenta anche la conoscenza dell'immagine.

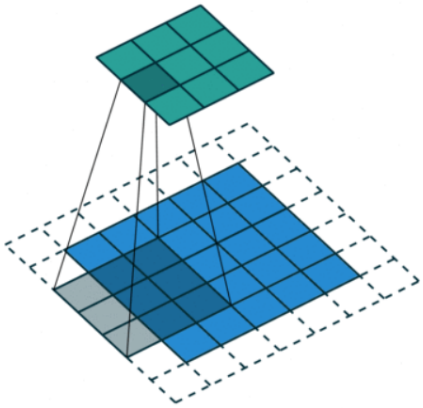


Figura 3.4: Convoluzione con padding

Ci sono due risultati differenti nei confronti dell'operazione:

- **Valid Padding**: in cui il risultato della convoluzione è ridotto rispetto alle dimensioni dell'input
- **Same Padding**: in cui la dimensionalità rimane invariata o è maggiore rispetto all'input

Successivamente, il *pooling layer* riduce la dimensione. Questo, come detto, abbassa la potenza computazionale necessaria per processare i dati. Generalmente vengono utilizzate due tipi di operazioni:

- **Average Pooling**: restituisce il valore medio di tutti i valori della porzione di immagine coperta dal *kernel*
- **Max Pooling**: restituisce il massimo valore della porzione di immagine coperta dal *kernel*

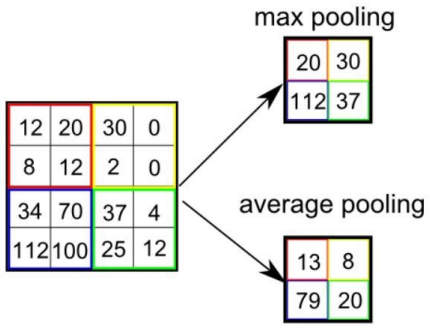


Figura 3.5: Esempio di pooling

Insieme, *Convolutional layer* e *Pooling layer*, formano uno dei livelli di CNN. Generalmente, esiste un *trade-off* fra complessità della rete, che permette di catturare anche dettagli di "basso livello", e potenza computazionale. Successivamente la matrice di uscita viene appiattita in un vettore monodimensionale colonna e questo costituisce l'ingresso della rete classificatrice. Se si aggiunge poi un livello *fully connected* è possibile apprendere combinazioni non lineari dei dettagli di alto livello.

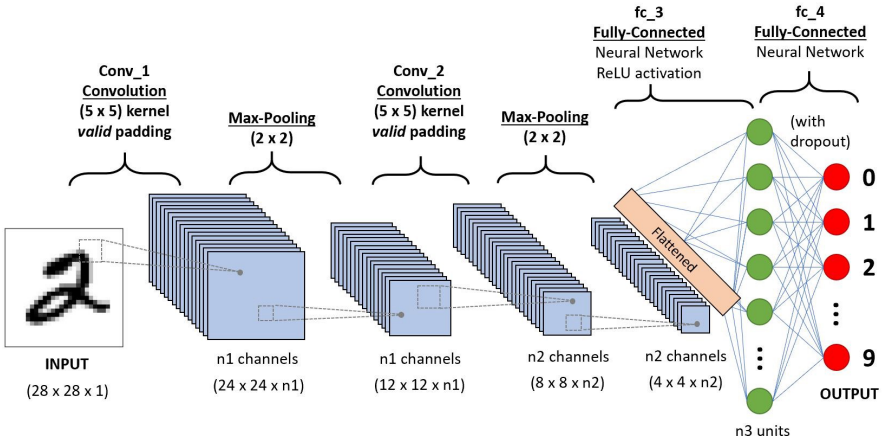


Figura 3.6: Architettura di CNN su un' immagine di MNIST

Capitolo 4

Implementazioni degli algoritmi sul dataset

4.1 Implementazione di GMM

Fino ad ora è stata fornita una illustrazione in forma generale rispetto a come funziona l'algoritmo utilizzato per la *Background Subtraction*. Di seguito si intende specificare il punto di partenza del lavoro e l'implementazione attuata.

Si è utilizzato, su ambiente Python, una libreria di funzioni apposite per *image processing*: OpenCV [19]. Essa permette all'utente di utilizzare molti algoritmi di processazione digitale delle immagini, garantendo una vasta gamma di funzioni con parametri modificabili. Una volta effettuata l'estrazione del *foreground*, la maschera in uscita viene ulteriormente filtrata al fine di rimuovere imperfezioni su regioni di pixel ambigue. La soluzione, nel particolare, per eliminare questi residui indesiderati è data dall'applicazione di operazioni morfologiche [20]. Diventa quindi necessario definire queste operazioni, che caratterizzano il codice. Esistono

diverse operazioni per eliminare le imperfezioni, ma tutte seguono un procedimento molto simile. Per prima cosa, si deve introdurre lo *structuring element*, un insieme di pixel usato come sonda per scansionare l'immagine. Esso può avere dimensioni e forme differenti, che cambiano a seconda dell'obiettivo da ottenere. In particolare, nel codice vengono utilizzate quattro operazioni morfologiche:

- **Apertura**: sia data un'immagine binaria I all'interno della quale sia presente un set A di pixel appartententi al *foreground*, su quale si vuole effettuare un'operazione di apertura tramite uno *structuring element* B. Ora si sovrapponga all'immagine I lo *structuring element* B. L'apertura di A da parte di B, indicata con il simbolo , si ottiene effettuando un'erosione di A, seguita da una dilatazione del risultato. In formula:

$$\boldsymbol{AB = (A \ominus B) \oplus B}$$

Figura 4.1: A sinistra l'immagine originale, a destra il risultato dell'apertura

- **Chiusura**: sia data un'immagine binaria I all'interno della quale sia presente un set A di pixel appartenenti al *foreground* sul quale si vuole effettuare una operazione di chiusura tramite uno *structuring element* B. Ora si so-

vrapponga all'immagine I lo *structuring element* B. La chiusura di A da B, indicata con il simbolo $\cdot$, si ottiene effettuando una dilatazione di A da parte di B, e poi un'erosione del risultato da B. In formula:

$$\boldsymbol{A \cdot B = (A \oplus B) \ominus B}$$

Figura 4.2: A sinistra l'immagine originale, a destra il risultato della chiusura

Alcune di queste operazioni morfologiche risultano necessarie per eliminare effetti indesiderati sulle immagini in uscita. In particolare nell'algoritmo di GMM si ricorre all'utilizzo di operazioni di chiusura ed apertura, al fine di correggere imperfezioni visibili nei frame su cui è stata fatta la segmentazione. Viene per questo scopo, utilizzata la funzione *cv2.morphologyEx* sulla maschera generata. Tornando al codice utilizzato, in un primo momento è stato creato un video a partire dalle immagini singole del *dataset*. A questo punto viene richiamata la funzione contenuta nella libreria di OpenCV *cv2.createBackgroundSubtractorMOG2*, dove *MOG2* [21] è un tipo di metodo per *background subtraction* a mistura di gaussiane, che segue l'algoritmo di GMM [17]. La funzione ritorna il modello di *background* a cui applicare il frame corrente attraverso il metodo *apply(frame)*, per generare

la maschera di *foreground*. Successivamente viene *"rifinita"* l'immagine ottenuta, sfruttando lo *structuring element*, identificato come *kernel*. Il filtro è una matrice (3×3), con una morfologia elissoidale. Dopo, sulla maschera così generata, vengono applicate tre operazioni quali: il *blur gaussiano*, un'operazione morfologica di *apertura* ed una di *chiusura*. Tutte e tre concorrono a migliorare il risultato finale. *Cv2.connectedComponentsWithStats* restituisce l'immagine etichettata, il numero delle etichette ed un *array* con le coordinate delle *bounding box*.

Algorithm 1 GMM pseudo-algoritmo

Input: dataset D formato da N immagini
Output: N frame di background subtraction del dataset D

repeat
    (Ri)creazione di un kernel ellittico;
    (Ri)creazione del *subtractor* MOG2;
    Applicazione filtro gaussiano;
    Applicazione apertura e chiusura;
    Incremento indice k;
until $k = N$;
Salvataggio delle immagini segmentate;

4.2 Implementazione di U-Net

U-Net [22] è la rete neurale utilizzata per la tesi, allo scopo di essere confrontata con gli algoritmi più classici di *computer vision*. Essa, è un esempio di *Convolutional Neural Network*. Questa architettura è stata utilizzata, in primo luogo, per problemi riguardati le immagini mediche, in particolare per svolgere compiti di classificazione. Successivamente, l'area di utilizzo di questa rete si è ampliata ed attualmente è usata anche per scopi di *Image Segmentation*, in cui il frame è diviso

in un segmento che rappresenta l'oggetto ed uno che invece indica il *background*. L'idea principale, a supporto di U-Net, è convertire inizialmente l'immagine in un vettore, per acquisire informazioni attraverso le *feature map*. Questo processo viene poi eseguito al contrario, utilizzando le mappature per ricostituire l'immagine. La rete può essere divisa in una parte che effettua l'*encoding*, seguita da un segmento che effettua il *decoding*. Il nome dell'architettura è dovuto alla sua particolare struttura ad "U". U-Net è formata da tre sezioni:

- sezione di **contrazione**: costituita da più blocchi di contrazione. Ogni blocco applica all'immagine in input due livelli di convoluzione, seguiti da due livelli di *max pooling*.
- sezione a **collo di bottiglia**: è una sezione che media fra la sezione di contrazione e quella di espansione.
- sezione di **espansione**: costituita da più blocchi di espansione. Il numero di blocchi di espansione è uguale al numero di blocchi di contrazione.

Il calcolo della funzione di perdita viene effettuato utilizzando specifici pesi. Nello specifico, vengono adottati pesi maggiori sui bordi degli oggetti segmentati.

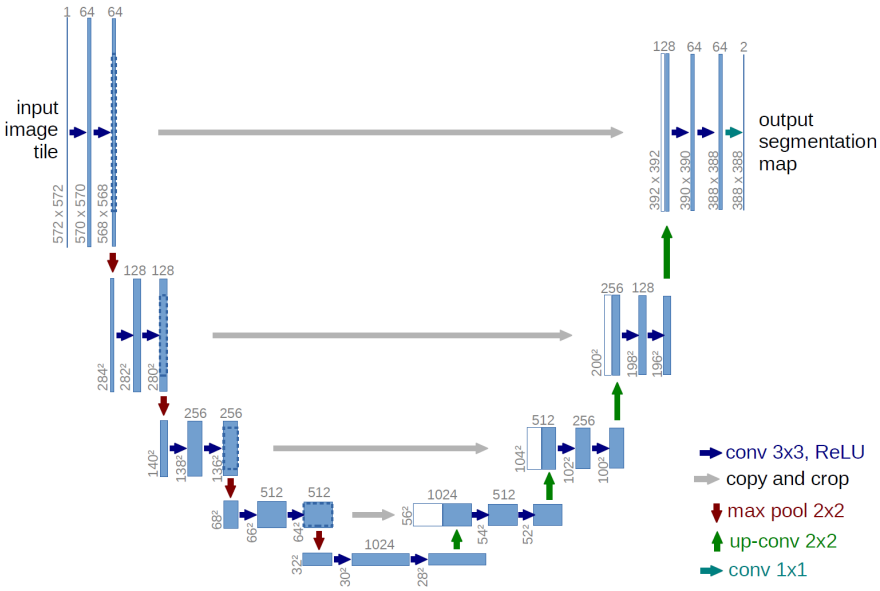


Figura 4.3: Esempio dell'architettura di U-Net

La rete tratta il problema di *image segmentation* come un compito di classificazione. In questo modo la funzione di perdita di U-Net ottiene delle migliori prestazioni rispetto a quelle tradizionali di altre architetture. Si è scelto nuovamente l'ambiente Python. In questo caso, assieme a OpenCV è stato usata la libreria Keras, ricca di funzioni specifiche per il funzionamento delle reti neurali. L'architettura è stata notevolmente semplificata a seguito di risultati comunque soddisfacenti. In particolare tramite la funzione *getModel* è stata costruita l'architettura, formata da cinque *layer*:

- il livello di input
- tre livelli di convoluzione 2D in sequenza
- il livello di output

La rete riceve quindi in ingresso le immagini ed esse passano, appunto, attraverso tre *convolutional layer*, espressi dalla formula *Conv2D*. Ogni livello di convoluzione effettua *same padding*, quindi l'immagine, a differenza di quanto accade in U-Net, mantiene sempre la stessa dimensione. Ognuno di questi tre livelli è seguito dalla funzione di attivazione ReLU. Infine, il livello d'uscita termina con la funzione di attivazione sigmoide. Come anticipato, si è reso necessario aumentare il numero di dati a disposizione. Specificatamente per questo, ciascuna immagine è stata ruotata lungo l'asse verticale. La stessa operazione è stata ovviamente effettuata anche per i frame di target, che rappresentano la *Background Subtraction*, ottenuti tramite un algoritmo che sfrutta la differenza assoluta con una soglia. Il *dataset* è stato, in seguito, diviso in una parte usata per l'apprendimento ed un'altra utilizzata invece per la fase di test.

Algorithm 2 CNN pseudo-algoritmo

Input: dataset D formato da N immagini; N immagini di target T, Background Subtraction delle immagini in input
Output: N frame di predizione di Background Subtraction del dataset D
repeat
    Rotazione sull'asse verticale della *k-esima* immagine di D e T;
    Incremento k;
until $k = D, T$;
Inizializzazione dei parametri del modello;
Creazione della struttura del modello e compilazione;
Validazione sul *train set*;
Validazione sul *test set*;

Capitolo 5

Analisi dei risultati sul dataset

5.1 Analisi dei risultati di GMM sul dataset

In merito alle prestazioni, GMM ha una complessità proporzionale al numero di gaussiane utilizzate, ma risulta fortemente robusto alle variazioni del *background*. In riferimento al *dataset* selezionato, come già anticipato, il numero di oggetti su cui effettuare la *Background Subtraction* aumenta con lo scorrere delle immagini. L'algoritmo riesce ad identificare la quasi totalità degli oggetti, a prescindere dalla loro inclinazione rispetto alla camera. Inoltre, esso risulta insensibile a zone con scarsa illuminazione e basso contrasto. Infine, la scelta dell'algoritmo di estrazione delle componenti connesse, tramite la funzione *cv2.connectedComponentsWithStats*, riguarda la sua successiva implementazione sulla scheda a microcontrollore. In ottica di riproduzione dell'algoritmo, su una piattaforma con risorse limitate, la strada delle componenti connesse risulta essere più veloce in termini di tempo di esecuzione rispetto a quella del tracciamento dei contorni.

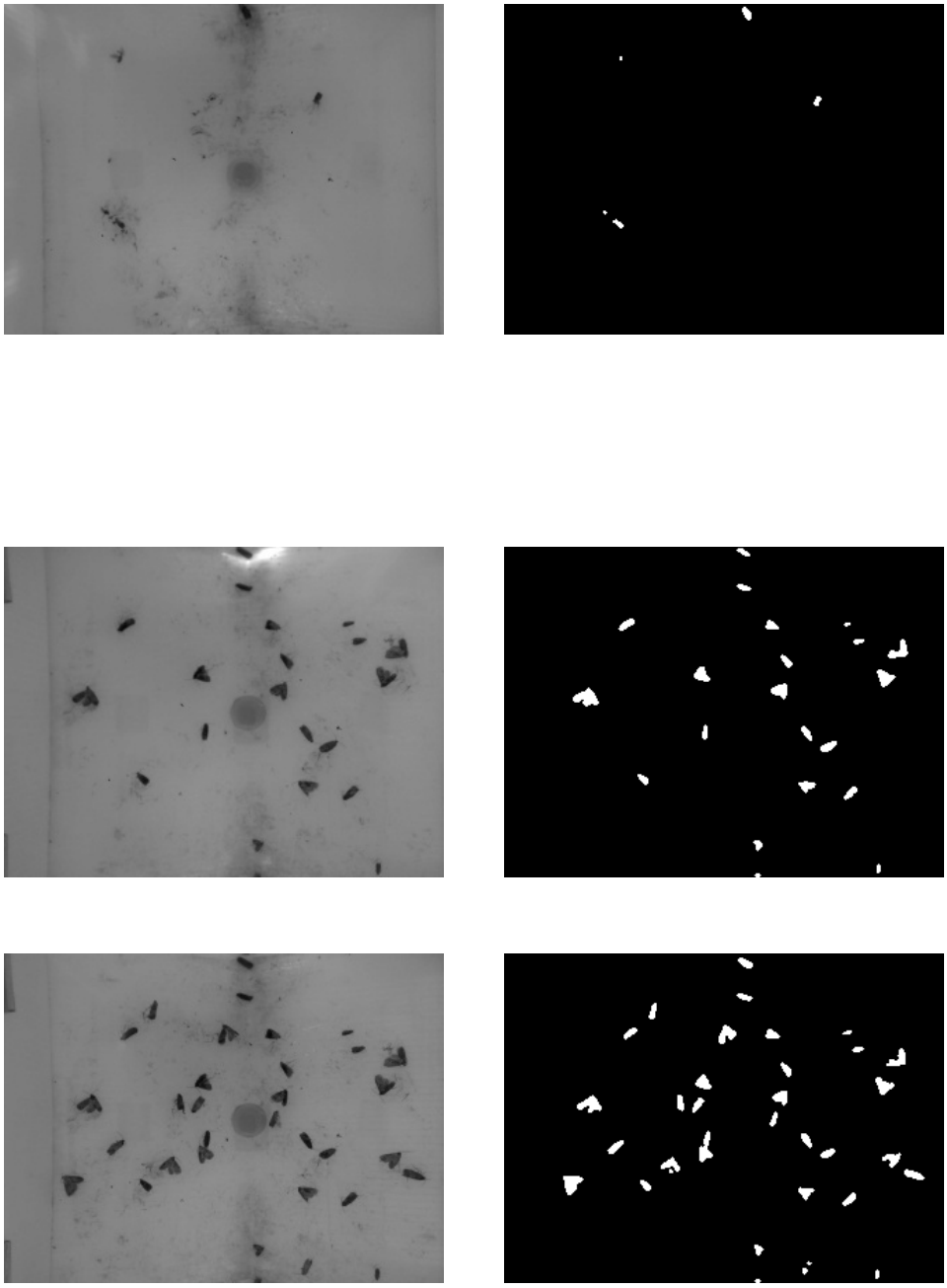
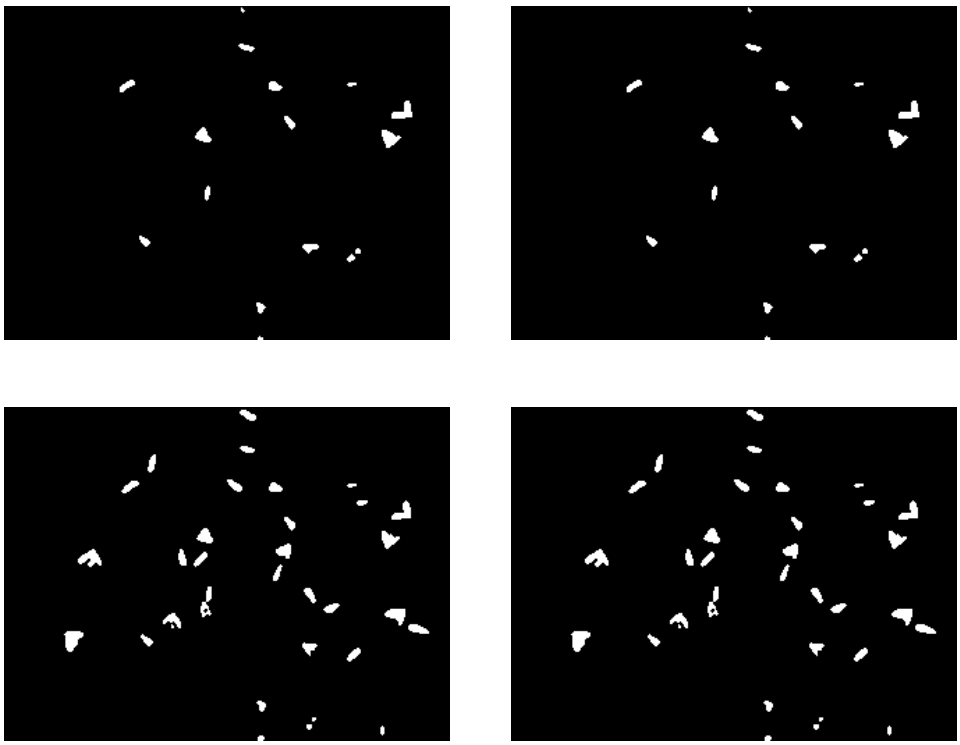


Figura 5.1: Esempi di Background Subtraction tramite GMM

5.2 Analisi delle prestazioni della CNN sul dataset

In riferimento al *dataset* selezionato, l'algoritmo basato su reti neurali presenta un alto grado di accuratezza, che verrà specificato successivamente. Il modello riesce ad identificare gli oggetti della scena, anche in questo caso, dopo variazioni di luce e inclinazioni rispetto alla camera. Di seguito vengono proposti alcuni esempi delle predizioni, confrontate con i risultati di GMM.



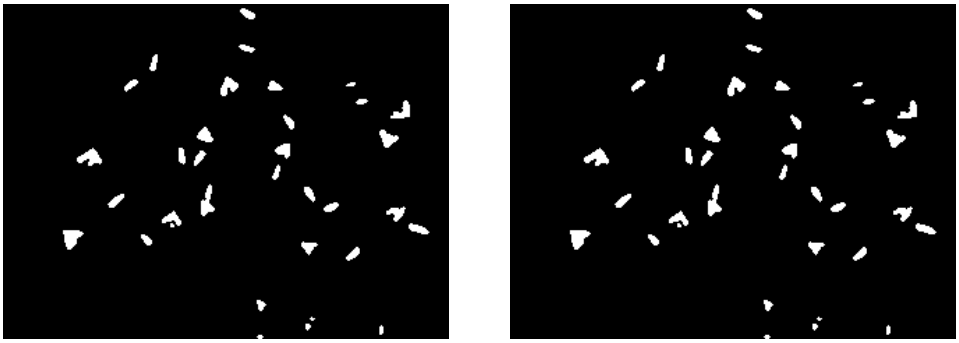


Figura 5.2: tre coppie di frame su cui sono state effettuate background subtraction: a sinistra le immagini ottenute attraverso GMM, a destra quelle ricavate dalla rete

Vengono studiate le proprietà degli algoritmi per valutare le prestazioni degli stessi. Il primo criterio è più grossolano ed indica il numero di errori, ossia il numero di oggetti erroneamente non identificati come *foreground*, oppure il numero di parti di *background*, classificate come oggetti. I dati calcolati, sono riportati nella seguente tabella:

	GMM	U-Net
Falsi Negativi	2	2
Falsi Positivi	3	1

Il secondo criterio si basa invece sulla precisione ottenuta calcolando la differenza nei valori dei singoli pixel che compongono le immagini. I valori riscontrati sono elevati, in un intervallo che si attesta fra il 98% ed il 99%. Queste valutazioni suggeriscono che il modello è estremamente specifico per il *dataset* scelto, ma ha scarse proprietà di generalizzazione. La principale causa è il basso numero di dati in ingresso con cui allenare il modello e non dipende, almeno in larga parte, dai parametri e dalla profondità della rete. Di seguito sono riportate le prestazioni del modello neurale nei confronti di GMM:

	Corrispondenza maggiore	Corrispondenza peggiore	Valor medio
Pixel Difference	99.9%	98.1%	99.3%

Capitolo 6

Conclusioni

La tesi ha proposto un confronto tra strategie "classiche" di *Background Subtraction* e tecniche basate su reti neurali. In particolare, sulla base degli studi precedentemente svolti, è stato definito un contesto per la realizzazione di due principali algoritmi di *computer vision*. Dopo una revisione della letteratura in merito a questo campo di studio, si è proceduto con l'implementazione di questi:

- un algoritmo più tradizionale, basato su *Gaussian Mixture Model*
- un algoritmo appartenente alla classe delle *Convolutional Neural Network*, ed in particolare derivato dall'architettura denominata U-Net

Poi, è stata presentata un'analisi di confronto tra le complessità delle due diverse tecniche.

L'utilizzo di algoritmi basati su GMM e "soglia e differenza" ha permesso di ottenere ottimi risultati per il *dataset* specifico, utilizzato per la dissertazione. Risulta ragionevole pensare, che una variazione dei dati in ingresso, in termini di variazioni di luci, oggetti in scena e prospettiva, restituirebbe risultati analoghi.

Differentemente, sono pochi i parametri all'interno del codice modificabili. Gli algoritmi tradizionali sono specifici per una classe di dati ed ottengono risultati simili per ogni immagine dello stesso insieme. Infine, l'utilizzo di un *dataset* così scarno, non ha influenzato le prestazioni degli algoritmi. Per la natura della tecnica, non si è reso necessario utilizzare *data augmentation*, ossia il processo con il quale si aumenta il numero di dati a disposizione, tramite modifiche alle immagini già presenti, oppure la sintesi di nuove, dello stesso tipo di quelle appartenenti al *dataset*.

L'algoritmo basato su una semplice rete neurale ha permesso di ottenere soddisfacenti predizioni per il *dataset* analizzato. Si è rivelato fondamentale aumentare il numero di dati a disposizione, questo ha permesso alla rete di ottenere abbastanza immagini su cui effettuare la fase di appredimento. Le reti neurali sono allenate, piuttosto che programmate; questo evita di dover affrontare precisi aggiustamenti nel codice. Esse inoltre, possono subire in seguito, un nuovo processo di apprendimento, su un altrettanto diverso insieme di dati. Di fatto quindi, gli algoritmi di reti neurali permettono una maggiore generalizzazione dell'implementazione. Risulta altrettanto vero che non è sempre ottenibile la generalizzazione del modello, nei confronti di qualsiasi tipo di insieme di dati. Concretamente, una rete neurale che trae eccellenti prestazioni in alcuni casi specifici, può non riuscire a performare con un set di dati reali. Inoltre, queste tecniche risultano eccessive, nel caso in cui l'applicazione sia rivolta ad uno specifico problema, in cui anche gli algoritmi come GMM ottengono risultati analoghi, con più semplicità d'implementazione. GMM è risultato essere l'algoritmo meno dispensioso, in termini di tempo e potenza computazionale. Di contro, un algoritmo basato sull'apprendimento, ottenuto tramite reti neurali, permette una maggiore generalizzazione ed una più alta efficacia nel

compito affidatogli. Si dimostra però necessario osservare che il funzionamento corretto di una rete neurale dipende da molteplici fattori, che riguardano non solo il tipo di dati, ma anche l'adeguatezza della rete e l'efficacia delle misure atte a predisporre il modello. Per concludere, seppur gli algoritmi basati sull'intelligenza artificiale rappresentino il più promettente e brillante paradigma per la *computer vision*, essi devono essere impiegati a seguito di un esame circa i requisiti e le criticità dell'applicazione.

Bibliografia

[1] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[2] Robert M Haralick and Linda G Shapiro. Image segmentation techniques. *Computer vision, graphics, and image processing*, 29(1):100–132, 1985.

[3] Oren Boiman, Eli Shechtman, and Michal Irani. In defense of nearest-neighbor based image classification. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

[4] Cewu Lu, Ranjay Krishna, Michael Bernstein, and Li Fei-Fei. Visual relationship detection with language priors. In *European conference on computer vision*, pages 852–869. Springer, 2016.

[5] Li Liu, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, and Matti Pietikäinen. Deep learning for generic object detection: A survey. *International journal of computer vision*, 128(2):261–318, 2020.

[6] Juan Du. Understanding of object detection based on cnn family and yolo. In *Journal of Physics: Conference Series*, volume 1004, page 012029. IOP Publishing, 2018.

[7] Massimo Piccardi. Background subtraction techniques: a review. In *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No. 04CH37583)*, volume 4, pages 3099–3104. IEEE, 2004.

[8] Rene Vidal, Joan Bruna, Raja Giryes, and Stefano Soatto. Mathematics of deep learning. *arXiv preprint arXiv:1712.04741*, 2017.

[9] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning*, pages 791–798, 2007.

[10] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[11] Antonia Creswell and Anil Anthony Bharath. Denoising adversarial autoencoders. *IEEE transactions on neural networks and learning systems*, 30(4):968–984, 2018.

[12] Chong Zhou and Randy C Paffenroth. Anomaly detection with robust deep autoencoders. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 665–674, 2017.

[13] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. In *2017 International Conference on Engineering and Technology (ICET)*, pages 1–6. Ieee, 2017.

[14] Hepu Deng. Multicriteria analysis with fuzzy pairwise comparison. *International journal of approximate reasoning*, 21(3):215–231, 1999.

[15] Ian Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016.

[16] Douglas A Reynolds. Gaussian mixture models. *Encyclopedia of biometrics*, 741:659–663, 2009.

[17] Zoran Zivkovic. Improved adaptive gaussian mixture model for background subtraction. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, volume 2, pages 28–31. IEEE, 2004.

[18] Xiaofan Lin, Cong Zhao, and Wei Pan. Towards accurate binary convolutional neural network. *arXiv preprint arXiv:1711.11294*, 2017.

[19] Alexander Mordvintsev and K Abid. Opencv-python tutorials documentation. *Obtenido de https://media. readthedocs. org/pdf/opencv-python-tutroals/latest/opencv-python-tutroals. pdf*, 2014.

[20] Ivan Culjak, David Abram, Tomislav Pribanic, Hrvoje Dzapo, and Mario Cifrek. A brief introduction to opencv. In *2012 proceedings of the 35th international convention MIPRO*, pages 1725–1730. IEEE, 2012.

[21] Tibor Trnovszkỳ, Peter Sỳkora, and Róbert Hudec. Comparison of background subtraction methods on near infra-red spectrum video sequences. *Procedia engineering*, 192:887–892, 2017.

[22] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.