Università Politecnica delle Marche

Dipartimento di Ingegneria Industriale e Scienze Matematiche

Laurea Magistrale in Ingegneria Meccanica - Meccanico Costruttivo

Tesi:

Realizzazione di un sistema di computer vision e deep learning per il controllo di qualità in linea di saldatura

Realization of a computer vision and deep learning system for soldering line quality control

Relatore:
Prof. Gian Marco Revel

Studente:
Enrico Ceresoni

2020-2021

Indice

1 Introduzione **5**

2 Reti Neurali **9**

2.1 Struttura delle reti neurali 11

2.2 Funzioni di attivazione 13

2.3 Inizializzazione e funzioni di perdita 17

2.4 Backpropagation e algoritmi di ottimizzazione per le reti neurali . 19

3 Metodi di elaborazione delle immagini **27**

3.1 Proprietà delle immagini digitali 28

3.2 Tecniche base di elaborazioni delle immagini 32

3.2.1 Point-Point Operation 32

3.2.2 Local Operators 34

3.3 Utilizzo delle reti neurali per elaborazione delle immagini 38

3.4 Metriche di valutazione della qualità delle immagini 41

3.4.1 Mean Square Error (MSE) 41

3.4.2 Structure Similarity Index Method (SSIM) 42

4 Valutazioni sui controlli di qualità **45**

5 Analisi preliminare della linea **51**

5.1 Layout linea automatica 51

5.2 Introduzione alla tecnica e processo di saldatura 53

5.3 Sistema attuale di controllo qualità della saldatura 56

5.3.1 Ispezione con machine vision 57

5.3.2 Ispezione visiva operatore 61

6 Sistema e metodologia di raccolta dati 63

6.1 Scelta del Hardware 64

6.2 Codice di Data Collection 67

7 Modello di Computer Vision 73

7.1 Algoritmo di misura 74

7.1.1 Data Preparation 74

7.1.2 Encoder 75

7.1.3 Decoder 76

7.1.4 Fase di Training 77

7.1.5 Metodo di misura 80

7.2 Classificazione 83

7.3 Performance rispetto agli altri sistemi di controllo 90

7.4 Incertezza di misura e taratura 92

8 Conclusioni ed Implementazioni future 97

Bibliografia 99

Capitolo 1

Introduzione

Durante l'esperienza di tirocinio in Cebi Italy ho avuto la possibilità di progettare un sistema di computer vision, che usasse delle logiche di deep learning capaci di riconoscere e misurare le non conformità nei giunti brasati con leghe di stagno.
Questo progetto nasce per valutare la possibilità di creare internamente alla azienda sistemi di controllo qualità con algoritmi di intelligenza artificiale che potessero interfacciarsi con la rete aziendale e monitorare in real time i processi, in questo caso, delle saldature robotizzate. Questo permetterebbe di prendere decisioni repentine per ripristinare le condizioni standard in caso di deriva del processo.
Lo studio inoltre si pone l'obiettivo di valutare l'efficacia del sistema di computer vision rispetto ai sistemi attualmente in linea (machine vision e ispezione visiva fatta dagli operatori), per valutare la possibilità di rendere questo nuovo metodo di controllo standardizzato e distribuito a tutti i processi di saldatura presenti in azienda.
Il progetto di tesi si è diviso in due macro-fasi, la prima di ricerca e sviluppo di un modello di deep learning e la seconda di implementazione in una linea automatizzata di saldatura per verificarne le prestazioni a confronto con gli attuali sistemi di ispezione visiva.
Nella fase iniziale, dopo aver analizzato il caso specifico, si è capito che si trattava di un problema a classi sbilanciate, ossia i difetti di saldatura sono in numero molto minore ai giunti conformi. In seguito alla lettura di diverse ricerche acca-

demiche e pubblicazioni scientifiche ([1],[2]), si è deciso di sviluppare un sistema di Anomaly Detection con reti neurali in grado di riconoscere e quantificare lo scostamento dallo standard di saldatura. Nella seconda parte dell'esperienza si è deciso con la direzione, di utilizzare una scheda di edge computing capace di permettere sia l'acquisizione dei dati utili per l'addestramento che per la valutazione degli algoritmi di deep learning (immagini dei giunti saldati ed esiti dei controlli qualità in linea mediante l'interfacciamento al PLC), che l'implementazione di quest'ultimi per verficarne in pratica le performance.
Attualmente il sistema di computer vision è integrato nella linea in modo passivo, ossia capace di acquisire segnali ed elaborarli con l'algoritmo sviluppato, ma senza rinviare il segnale al PLC e senza cambiare le condizioni operative dell'ispezione visiva fatta dall'operatore, in quanto ancora non testato a sufficienza.
Nel Capitolo 2 di questa trattazione si presenterà la teoria base delle reti neurali [3], spiegando la struttura del percettrone, le principali funzioni di attivazioni e come essi interagiscono tra di loro per creare una rete neurale. In seguito, si discuterà della fase di addestramento di quest'ultima partendo dall'inizializzazione dei suoi pesi fino alle tipologie di funzioni di perdità (loss function), utili per capire quanto il risultato predetto dal modello si discosta da quello reale. A conclusione del capitolo si è parlato delle tecniche di ottimizzazione per ridurre la funzione di perdita e quindi creare una rete che riesca ad approssimare al meglio la realtà.
Nel Capitolo 3 si è parlato delle tecniche e metodi di elaborazione delle immagini utili per la comprensione di questa trattazione([4][5]). Si sono viste in dettaglio le proprietà delle immagini digitali e i metodi di elaborazione Point Point Operation, Local Operation[6] e le reti neurali convoluzionali[3]. Alla fine, si sono presentate due metriche (MSE, SSIM)[7] per la valutazione della qualità dell'immagine.
Il capitolo 4 ha lo scopo di descrivere il processo di valutazione di un sistema di controllo industriale, presentando il concetto di FP (False Positive), FN (False Negative), TP (True positive) e TN (True Negative)[8] mediante strumenti come la Confusion Matrix e metriche quali il Recall, Precision e F1-score[9]. Si è introdotto anche il problema delle classi sbilanciate e come le metriche precedenti

sono state corrette per avere una maggior sensibilità nella valutazione del problema (Macro Recall, Macro Precision e Macro F1-score)[10].
Nel Capitolo 5 è stato presentato il contesto in cui si andrà a sviluppare il sistema di computer vision e quindi si è descritta la linea automatica ed ogni singola stazione, con particolare attenzione alle stazioni di saldatura automatizzate e ai controlli qualità in linea. Si è presentata la tecnica di saldatura utilizzata, ossia la brasatura dolce a Stagno, e il processo automatizzato che la produce specificando anche i possibili difetti che in genere si verificano. A seguire, si sono descritti i due metodi di ispezione delle saldature che si hanno in linea, ossia il sistema di machine vision e il controllo qualità effettuato dall'operatore.
Il Capitolo 6 si concentra sulla progettazione del sistema per la raccolta dati e lo sviluppo del codice necessario. Per l'hardware si è scelta una scheda di edge computing (Jetson Nano-B01 [11]) in grado di soddisfare il collegamento ai dispositivi di riferimento (microscopio industriale e PLC) e che abbia una capacità computazionale adeguata anche per l'implementazione finale del sistema di computer vision.
Nel Capitolo 7 si sono spiegati i 4 algoritmo gemelli di deep learning (Convolutional Autoencoder per Anomaly Detection) che il sistema di computer vision usa per estrapolare una misura della conformità per i 4 giunti saldati. Da questa misura è possibile effettuare la classificazione di conforme o non mediante la scelta appropriata delle soglie di ogni singolo giunto. Per scegliere la soglia corretta si è deciso di attuare un sistema di ottimizzazione combinatoria che potesse minimizzare gli errori commessi, ossia che massimizzasse il macro F1-score. In seguito, si sono presentati i risultati ottenuti da un campione di produzione e si sono confrontati con i risultati dei sistemi attuali in linea. Per concludere si è valutata l'incertezza del sistema di misura [12] e come si sono risolti i problemi sistematici di ricostruzione di particolari giunti saldati.
Nella Capitolo 8 si sono riassunti i risultati ottenuti e presentitati i possibili miglioramenti e prossimi sviluppi di questo progetto.
Per tutto il lavoro è stato usato il linguaggio Python, utilizzando i notebook Jupyter per lo sviluppo degli algoritmi di Deep Learning (Keras e TensorBoard) e per

la fase di analisi e visualizzazione dei dati (Pandas, Numpy, Scikit Learn, Matplotlib, Plotly), mentre per lo sviluppo dei codice nella scheda di edge computing si è utilizzato IDE Pycharm con le librerie Jetson.GPIO, OpenCV e Pandas.

Capitolo 2

Reti Neurali

Quando si parla di intelligenza artificiale e in particolare di Machine Learning non si sta parlando esclusivamente di reti neurali ma esistono altri algoritmi di analisi, per citarne alcuni, la Random Forest, che usa alberi decisionali per effettuare problemi di classificazione, oppure le Support Vector Machines (SVM), che sono basate sull'idea di trovare un iperpiano che divida al meglio un set di dati in due o più classi. In questa trattazione si analizzeranno esclusivamente le reti neurali.

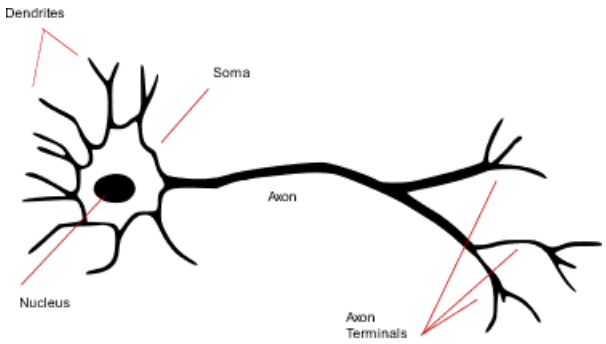


Figura 2.1: Struttura biologica del neurone

Le reti neurali nascono dall'emulazione del funzionamento biologico del neurone. Il neurone dai dendriti riceve input elettrici, il nucleo rielabora l'informazione e infine il risultato esce dall'assone, singolo output, che stimola altri dendriti di altri neuroni a valle (vedi Fig 2.1). Il corrispettivo neurone nelle reti neurali si chiama percettrone. Nella Fig 2.2 è mostrata la struttura del percettrone dato

un singolo elemento di input con x_n caratteristiche (features). In genere, una rete neurale per essere chiamata tale deve essere composta da più strati di percettroni, come il cervello umano che è composto da molti neuroni.

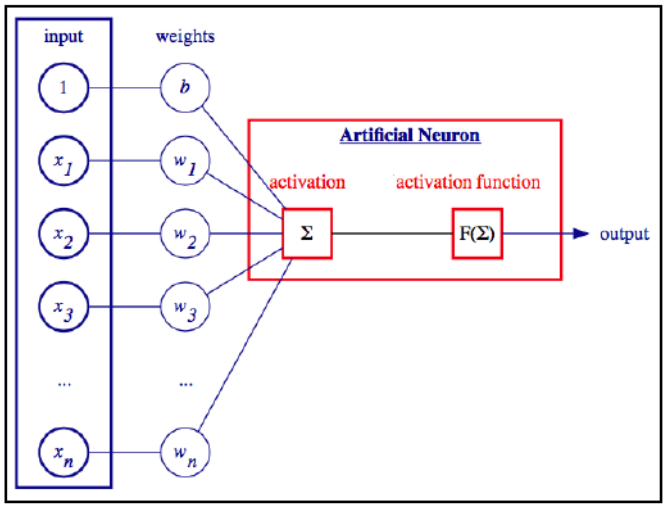


Figura 2.2: Struttura del percettrone

2.1 Struttura delle reti neurali

Nel paragrafo precedente si è introdotto il concetto del percettrone, come unità fondamentale delle reti neurali. A livello matematico il percettrone può essere espresso in forma esplicita come:

$$y = \Phi(\sum_i^n x_i w_i + b\ )$$

Dove con Φ si sta indicando la funzione di attivazione. In forma vettoriale come:

$$y = \Phi(\bar{w}^T \bar{x} + b)$$

Quindi l'output del percettrone y è la somma della combinazione lineare tra le caratteristiche dell'input $\bar{x}$ con un vettore $\bar{w}$ chiamato vettore dei pesi e con un coefficiente b detto bias. Il risultato ottenuto è inserito in una funzione chiamata funzione di attivazione Φ che rende la relazione non lineare. Quando si parla di reti neurali si parla di strati di percettroni, detti layers, concatenati tra di loro. Questi si dividono in 3 tipologie:

1. **Input layer:** Non è uno strato di percettroni. Rappresenta il vettore delle caratteristiche di input che verranno passate al primo strato della rete neurale.

2. **Hidden layers:** Sono gli strati intermedi della rete neurale. Il numero di percettroni di ogni singolo layer e il numero di layer viene determinato dallo sviluppatore nella fase iterativa di ricerca del modello. All'aumentare degli strati e dei percettroni per ogni strato aumentano i parametri da allenare della rete e quindi il costo computazionale per l'addestramento diventa importante. Quest'ultimo aumento, in genere, corrisponde un miglioramento delle prestazioni delle reti neurali. Ci sono dei casi, in cui reti troppo profonde soffrono dell'esplosione del gradiente (exploding gradients) che portano ad un peggioramento delle prestazioni ed è possibile risolverlo con i blocchi residui (residual blocks). Questa parte è stata citata solo per completezza ma in questa trattazione non verrà approfondito questo argomento.

3. **Output layer:** E' l'ultimo strato di percettroni che restituisce l'output della rete. Per problemi di classificazione binaria l'ultimo strato presenta un solo percettrone in quanto si avrà un esito binario, e in genere come funzione di attivazione si ha la sigmoide. Per problemi multiclasse si avrà tanti percettroni quante sono le classi da predirre e come funzione di attivazione si utilizzerà quasi sempre la softmax, che restituisce ad ogni elemento analizzato la probabilità di appartenenza ad ogni singola classe.

A livello matematico se si parla di strati di percettroni, il vettore dei pesi $\bar{w}$ diventa una matrice dei pesi W e il coefficiente bias b diventa un vettore $\bar{x}$. La formula matriciale generica diventa:

$$\bar{h}_{l+1} = \Phi(W_{l+1}^T \bar{h}_l + \bar{b}_{l+1})$$

Dove l rappresenta il numero del layer di riferimento che varia da 0 (input layer) a N (output layer). Per spiegare meglio il concetto, si sono ricavate tutte le equazioni in gioco per ogni layer della rete neurale in Fig 2.3:

Layers	**Equazione**	**Dimensioni** W	**Dimensioni** b
layer 1	$\bar{h}_1 = \Phi(W_1^T \bar{x} + \bar{b}_1)$	$W_1 = dim(3,5)$	$\bar{b}_1 = dim(3,1)$
layer 2	$\bar{h}_2 = \Phi(W_2^T \bar{h}_1 + \bar{b}_2)$	$W_2 = dim(3,3)$	$\bar{b}_2 = dim(3,1)$
layer 3	$y = \Phi(\bar{w}_3^T \bar{h}_2 + b_3)$	$\bar{w}_3 = dim(3,1)$	$b_3 = dim(1,1)$

Dalle equazioni precedentemente descritte, si capisce come aumentando i layer e/o il numero di percettroni in ogni singolo layer, la dimensione della matrice W e del vettore $\bar{b}$ aumentano e con loro il totale dei parametri che in fase di addestramento la rete deve ottimizzare. Questo porta ad un costo computazionale elevato che si traduce in maggior tempo di addestramento.

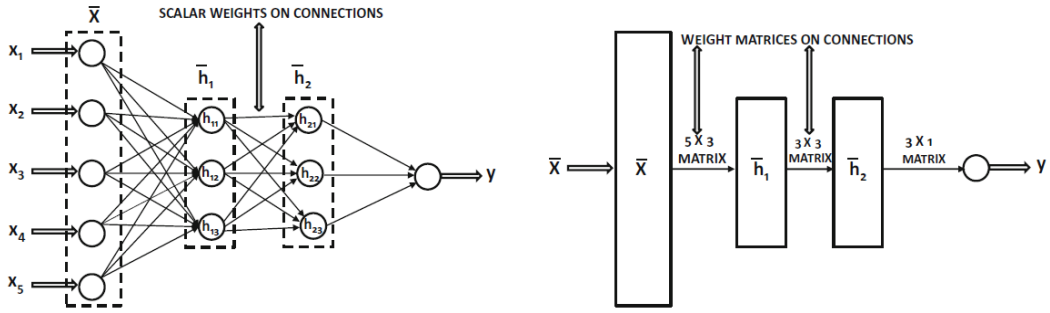


Figura 2.3: A sinistra si ha la rappresentazione scalare di una semplice rete neurale. A destra si ha la notazione vettoriale.

2.2 Funzioni di attivazione

Si è parlato spesso di funzione di attivazione, ossia quella funzione che trasforma una combinazione lineare, chiamato anche pre activation value, in un risultato non lineare chiamato post activation value (Fig 2.4).

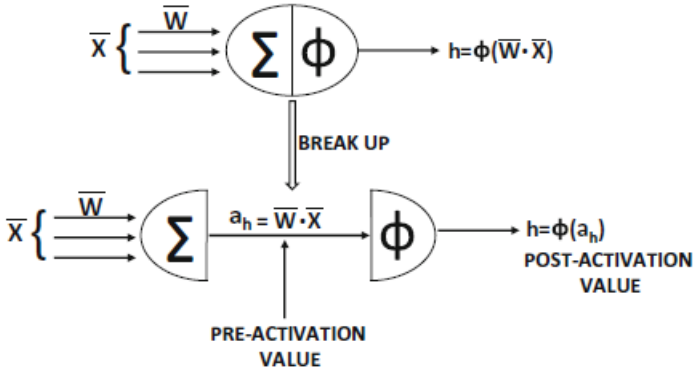


Figura 2.4: Percettrone

Ci sono diverse tipologie di funzione di attivazione ed ognuna possiede delle caratteristiche. Scegliere una funzione di attivazione rispetto ad un'altra potrebbe portare a grosse differenze a livello di efficienza della rete neurale, sia in fase di addestramento che in fase di esecuzione. Ad oggi le funzioni di attivazioni presenti in campo applicativo sono:

1. **Sigmoide:** Questa funzione restituisce sempre un valore compreso tra 0 e 1 (Fig 2.5). E' molto usata per effettuare sia regressioni che classificazione binaria nel ultimo strato della rete neurale.

$$\Phi(a) = \frac{1}{1 + e^{-a}}$$

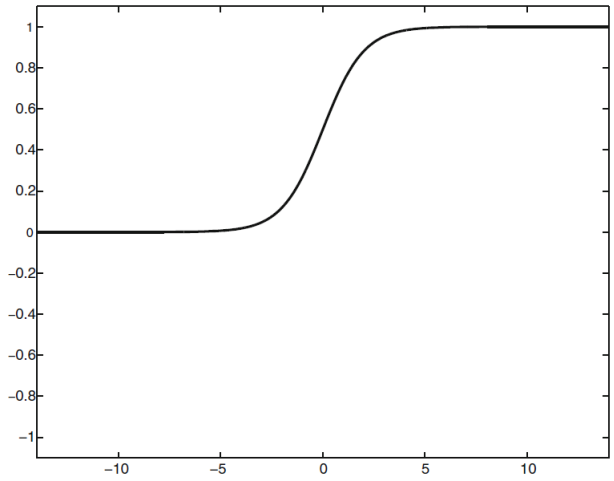


Figura 2.5: Sigmoide

2. **Tangente iperbolica**(*tanh*): Funzione molto simile alla sigmoide con la differenza che l'output è sempre compreso tra 1 e -1 (Fig 2.6).

$$\Phi(a) = \frac{e^{2a} - 1}{e^{2a} + 1}$$

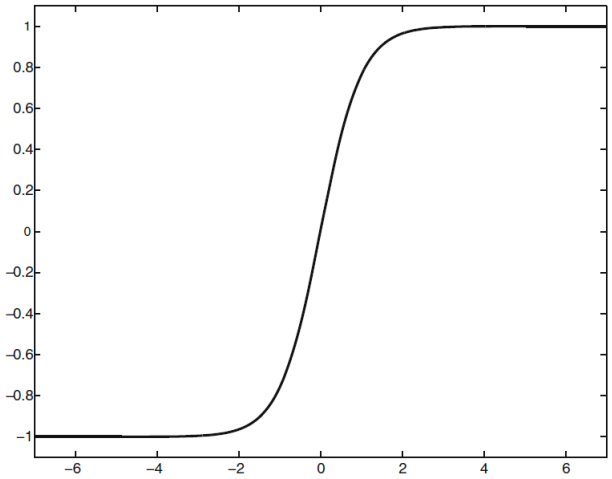


Figura 2.6: tanh

3. **ReLU**(Rectified Linear Unit): Questa è la funzione di attivazione maggiormente utilizzata nell'ambito delle reti neurali in particolar modo, nelle reti convoluzionali e nel deep learning, in quanto molto più efficiente dal punto di vista del costo computazionale. Per come è stata sviluppata, per tutti i valori negativi del pre-activation value, il valore di output sarà 0, se positivo invece restituisce il valore corrente (Fig 2.7). Da quello detto precedentemente si capisce che la maggior efficienza computazionale è dovuta dalla disattivazione di percettroni con valori negativi.

$$\begin{cases} \Phi(a) = a & \text{se } a > 0 \\ \Phi(a) = 0 & \text{se } a =< 0 \end{cases}$$

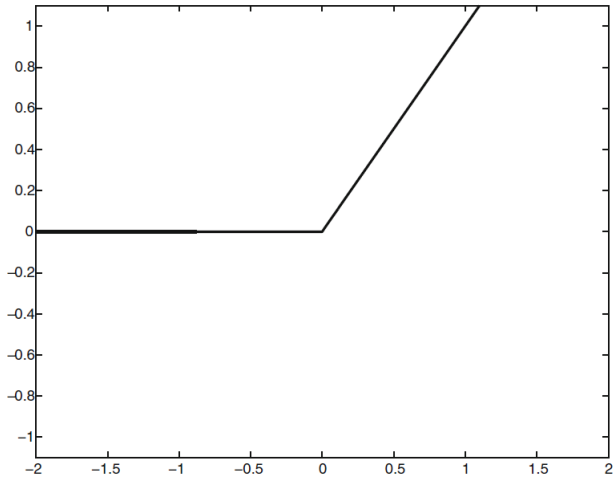


Figura 2.7: ReLU

Negli ultimi anni si sono introdotte nuove funzioni di attivazioni che provano a perfezionare la ReLU, in quanto quest'ultima potrebbe portare a dei percettroni che ripetutamente non vengono attivati e quindi anche i pesi e il bias relativo non verrà aggiornato (Dying ReLU Problem). Una funzione che risolve in parte questo problema è la Leaky ReLU, che introduce una piccola pendenza nella parte negativa, in genere 0.01 (Fig 2.8).

$$\begin{cases} \Phi(a) = a & \text{se } a > 0 \\ \Phi(a) = 0.001a & \text{se } a =< 0 \end{cases}$$

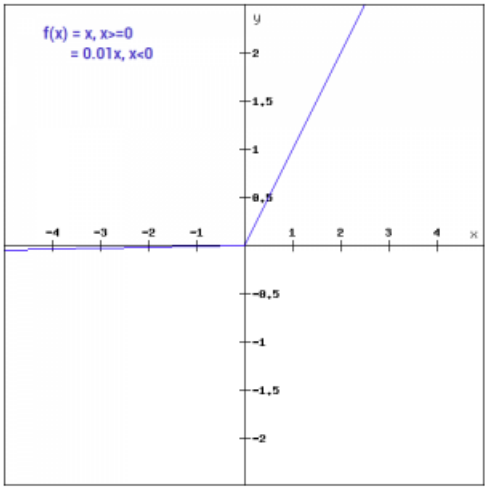


Figura 2.8: Leaky ReLU

Un'altra funzione di attivazione molto simile alla precedente è la ELU (Exponential Linear Unit). La differenza tra la ReLU sta nella parte negativa, dove la ELU presenta una curva esponenziale (Fig 2.9). Questa funzione di attivazione performa molto meglio rispetto alle precedenti in fase di addestramento, mentre in fase di testing risulta leggermente più lenta rispetto ad una rete addestrata con la ReLU.

$$\begin{cases} \Phi(a) = a & \text{se } a > 0 \\ \Phi(a) = \alpha(e^a - 1)a & \text{se } a =< 0 \end{cases}$$

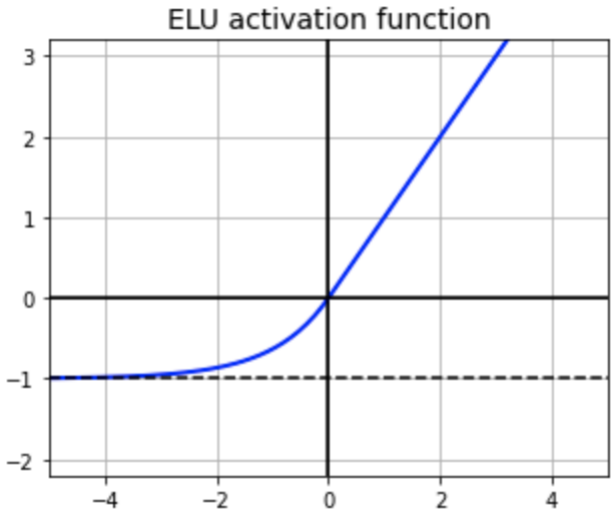


Figura 2.9: ELU

Il coefficiente α è un parametro che serve per determinare il limite inferiore negativo di output, in genere $\alpha = 1$.

2.3 Inizializzazione e funzioni di perdita

Una volta nota la struttura generale delle reti neurali e delle sue funzioni di attivazioni, si possono introdurre i concetti forward propagation e backward propagation, il concetto della funzione di perdita (loss function) e del metodo di ottimizzazione. L'apprendimento in una rete neurale corrisponde ad un problema di ottimizzazione. In prima fase, si esegue la fase di inizializzazione dei pesi e dei bias. Questa inizializzazione potrebbe essere randomica o, come in genere si usa, scelta dalla letteratura. In questi ultimi anni ci sono stati molti studi su quale sia l'inizializzazione migliore dei pesi e dei bias, in quanto una buona inizializzazione riduce il tempo di addestramento e non ci sarà il rischio di incorrere ne, nella problematica dell'esplosione del gradiente ne nella scomparsa del gradiente. L'inizializzazione più comune in letteratura e che si usa nella pratica è quella di He, in quanto migliore per le reti neurali con funzione di attivazione ReLU, sennò esiste anche l'inizializzazione di Xavier, che funziona bene con la tanh. Non si presenteranno approfondimenti matematici a riguardo, in quanto non è lo scopo di questa trattazione.
La seconda fase è quella della forward phase. In questo step, la rete neurale, con i pesi e i bias inizializzati, riceve in input i dati di training che a sua volta attivano a cascata tutti i percettroni degli hidden layers. L'output finale della rete neurale è un valore/esito che verrà confrontato con il valore/esito noto degli input mediante una funzione detta funzione di perdita. Questa funzione di perdita o loss function, che si indicherà con L, restituisce un valore chiamato perdita e maggiore è questa perdita maggiore sarà l'errore che commette la rete neurale nella predizione.
Le funzioni di perdita che si possono scegliere sono molteplici e, come nelle funzioni di attivazione, la scelta di una al posto che di un'altra incide sull'addestramento della rete neurale. In genere le funzioni di perdita si dividono per il campo ap-

plicativo, ossia per la regressione e per la classificazione.
Se siamo nel caso della regressione la funzione di perdita utilizzata è l'errore quadratico medio o MSE.

$$L = MSE = \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2$$

Dove m è il numero di elementi passati alle reti neurali, quindi avremo m soluzioni predette ($\hat{y}$) confrontate con m soluzioni reali (y). In sintesi, MSE calcola la media delle differenze al quadrato tra i valori previsti ed i valori reali. Il risultato sarà sempre positivo e tanto più il valore del MSE è prossimo a zero tanto più i valori predetti sono simili ai valori reali.
Nel caso di problemi di classificazione si usa l'entropia incrociata o Cross-Entropy. L'entropia incrociata è una misura della differenza tra due distribuzioni di probabilità per una data variabile casuale o insieme di eventi. Andando proprio sul contesto delle reti neurali, possiamo distinguere una distribuzione di probabilità vera P e una distribuzione di probabilità degli esiti predetti Q. Quindi l'entropia incrociata dipenderà dalle due distribuzioni:

$$L = H(P, Q) = -\sum_y P(y) * \log Q(y)$$

Dove x rappresenta ogni singola classe. Per esempio, in un problema di classificazione binaria la formula dell'entropia incrociata diventerebbe:

$$H(P, Q) = -(P(classe0) * \log(Q(classe0)) + P(classe1) * \log(Q(classe1))))$$

Questa viene chiamata anche Binary Cross-Entropy, in quanto specifica per la classificazione binaria. Nel caso multiclasse la formula esplicita diventerebbe:

$$H(P, Q) = -(P(classe0) * \log(Q(classe0)) + ... + P(classeX) * \log(Q(classeX))))$$

Questa viene chiamata anche Categorical Cross-Entropy.
Una volta scelta la funzione di perdita adeguata, va scelto come ultimo step il metodo di ottimizzazione, ossia l'algoritmo che permette alla rete neurale nella fase di backpropagation di aggiornare i propri pesi e bias in modo da ridurre la funzione di perdita.

2.4 Backpropagation e algoritmi di ottimizzazione per le reti neurali

La scelta dell'algoritmo di ottimizzazione è cruciale per il tempo di addestramento della rete e per trovare il minimo globale della funzione di perdita. Prima di parlare dei metodi di ottimizzazione, si spiegherà il sistema di backpropagation, ossia come calcolare il gradiente della funzione di perdita rispetto ad ogni matrice dei pesi e dei bias. Matematicamente parlando, il gradiente della funzione di perdita (∇L) rispetto alle variabili w_i, b_i, può essere scritto:

$$\nabla_w L = \left[\frac{\partial L}{\partial w_1}, \frac{\partial L}{\partial w_2}, ..., \frac{\partial L}{\partial w_m}\right]$$

$$\nabla_b L = \left[\frac{\partial L}{\partial b_1}, \frac{\partial L}{\partial b_2}, ..., \frac{\partial L}{\partial b_n}\right]$$

L'aggiornamento dei coefficienti w_i, b_i, avviene secondo tale formula:

$$w_{new} = w_{old} + \Delta w$$
$$b_{new} = b_{old} + \Delta b$$

Dove il ∇w e ∇b sono uguali a:

$$\Delta w = -\alpha \nabla_w L$$
$$\Delta b = -\alpha \nabla_b L$$

Il parametro α è chiamato learning rate ed è un iperparametro che ha la funzione di scalare il valore del gradiente. Quando si imposta un learning rate alto significa che si vuole dar molto valore al parametro di aggiornamento. In genere questo comporta in una fase iniziale una diminuzione consistente della loss function ma c'è il rischio di non riuscire ad andare a convergenza nel minimo globale della funzione, in quanto ho una minor sensibilità di aggiornamento e quindi il valore di perdita inizierà ad oscillare. Al contrario, nel caso di un learning rate basso l'aggiornamento dei coefficienti sarà più lento ma avremo una maggior sensibilità e quindi una probabilità maggiore di avvicinarsi al minimo globale della funzione di perdita (Fig 2.10). In genere i valori di α sono delle potenze di 10 e nella

pratica si usano 10^{-1}, 10^{-2} o 10^{-3}.

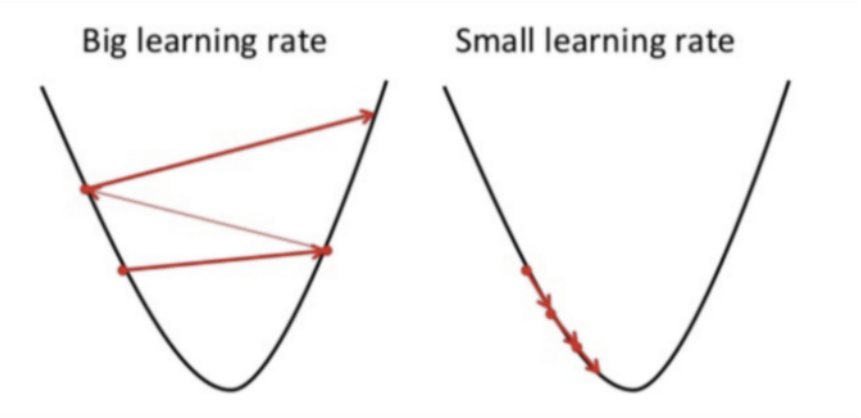


Figura 2.10: Comportamento del algoritmo di ottimizzazione in funzione al learning rate α

Quindi alla fine la formula di aggiornamento dei pesi e dei bias sarà:

$$w_{new} = w_{old} - \alpha \nabla_w L$$
$$b_{new} = b_{old} - \alpha \nabla_b L$$

Il passo successivo è capire come calcolare le derivate della funzione di perdita rispetto ad un singolo peso o bias, e per far ciò si deve introdurre il concetto della derivata di funzioni composte, chiamato chain rule. Data una funzione composta $f(g(x))$, la derivata di f rispetto a x sarà:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial x}$$

Questa è la formula della chain rule univariata.
Se la funzione è composta da due altre funzioni, cioè del tipo $f(g(x), h(x))$ la derivata di f rispetto a x sarà:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial x} + \frac{\partial f}{\partial h}\frac{\partial h}{\partial x}$$

Questa è la formula della chain rule multivariata.
Facendo riferimento alla formula generale della rete neurale è possibile capire come ogni equazione degli hidden layer sia collegato all'equazione dello strato precedente e così via. Quindi per trovare la derivata della funzione di loss rispetto ad un determinato peso (o bias) bisogna utilizzare più volte la tecnica della chain rule. La formula generica per trovare la derivata della loss rispetto ad un determinato peso è la seguente:

$$\frac{\partial L}{\partial w_{(h_{l-1},h_l)}} = \frac{\partial L}{\partial h_N}\left[\sum_{[h_l,h_{l+1},\dots,h_{N-1},h_N]} \frac{\partial h_N}{\partial h_{N-1}} \prod_{i=l}^{N-2} \frac{\partial h_{i+1}}{\partial h_i}\right] \frac{\partial h_l}{\partial w_{(h_{l-1},h_l)}}$$

$$\forall\ l\ \in\ [1, N]$$

Si può notare come il temine $\frac{\partial h_N}{\partial h_l} \prod_{i=l}^{N-2} \frac{\partial h_{i+1}}{\partial h_i}$ è la formula della chain rule univariata generale, mentre $\sum_{[h_l,h_{l+1},\dots,h_{N-1},h_N]} \frac{\partial h_N}{\partial h_{N-1}} \prod_{i=l}^{N-2} \frac{\partial h_{i+1}}{\partial h_i}$ è la formula della chain rule multivariata generale. Per non appesantire la trattazione si scriveranno esclusivamente le formule riguardo i pesi w, in quanto le formule sono matematicamente uguali per i bias b.
Il metodo di aggiornamento dei pesi presentato precedentemente è uno dei metodi che fa parte dei sistemi di ottimizzazione della discesa del gradiente e si chiama Batch Gradient Descent. Questo particolare algoritmo aggiorna i pesi una volta che ha calcolato il gradiente di ogni peso per tutti gli elementi del nostro dataset di addestramento. Questo sistema a livello computazionale è molto oneroso. Per prevenire a questo costo computazionale si è sviluppato un altro algoritmo di discesa del gradiente chiamato Stochastic Gradient Descent (SGD), che aggiorna i pesi ogni volta che viene calcolato il gradiente per ogni singolo elemento del dataset di training. Rispetto al precedente è molto più veloce e computazionalmente efficace ma comporta un aumento del rumore nella curva di apprendimento. La curva di apprendimento è un linechart dove nell'asse delle ascisse si hanno le epoche, ossia quante volte viene passato il dataset di addestramento alla rete, e

nell'asse delle ordinate si ha il valore della funzione di loss (esempio capito del progetto).
Il terzo ed ultimo algoritmo di discesa del gradiente è una sintesi dei due sistemi visti in precedenza ossia quello del Mini-batch Stochastic Gradient Descent. Questo metodo, mi permette di aggiornare i pesi dopo l'analisi di ogni singolo lotto di elementi del training dataset. Si intuisce che più elementi si avrà in ogni singolo lotto, minori saranno i lotti e quindi ci si avvicina al concetto del Batch Gradient Descent, mentre se si diminuiscono gli elementi per ogni lotto, aumentaranno il numero di lotti e quindi si avrà un comportamento simile algoritmo Stochastic Gradient Descent. In genere si preferiscono lotti di 8, 16, 32, 64, 128, 256... numeri che corrispondono alle memorie informatiche. Questa tecnica di ottimizzazione ha tre principali limiti:

1. Se la funzione di loss cambia lentamente in una direzione e velocemente in un'altra, potrebbe portare ad una grossa oscillazione del gradiente e quindi rallentare il processo di addestramento.

2. Se la funzione di loss ha un punto di sella, l'algoritmo potrebbe rimanere bloccato in questo "falso" minimo, in quanto il gradiente in questo caso risulta nullo e quindi non si avrà l'aggiornamento dei pesi.

3. Ancora si hanno gradienti rumorosi che potrebbero inficiare l'aggiornamento dei pesi

Per raggirare questi inconvenienti si è migliorato il sistema Mini-Batch SGD con quello che viene chiamato momento. Questo algoritmo riesce ad accelerare gli aggiornamenti dei pesi lungo la direzione in cui porta la funzione di loss a diminuire maggiormente. Viene definito come segue:

$$w_{new} = w_{old} - \alpha v_t$$
$$v_t = \beta v_{t-1} + (1 - \beta)\nabla_w L$$

dove v_t rappresenta la media mobile esponenziale dei valori dei gradienti. Come si può notare si è introdotto un altro iperparametro, il coefficiente β exponential

decay rate, che varia tra 0 e 1. Maggiore è il valore di β, ossia tendente a 1, maggiore sarà l'appiattimento del rumore in quanto si riduce l'importanza del valore dato dal gradiente, al contrario, minore è β minore sarà l'appiattimento del rumore (Fig 2.11).

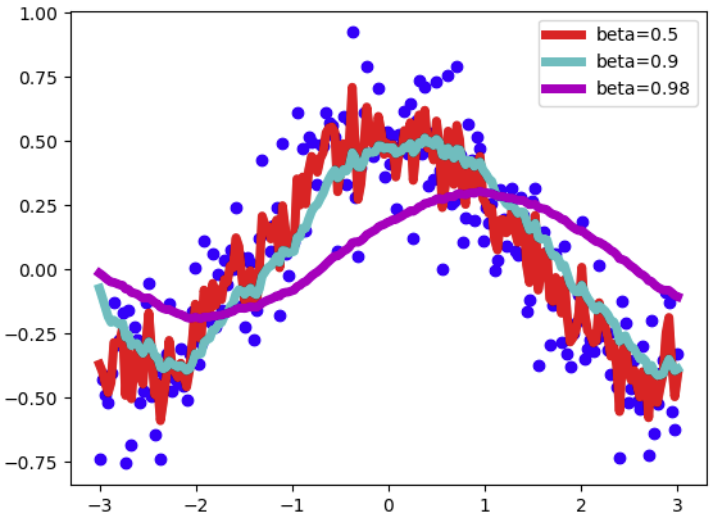


Figura 2.11: Effetti di β sulla media mobile

Si è visto a livello sperimentale che il valore β=0.9 è il valore migliore per il SGD con momento. In conclusione, rispetto al mini-batch SGD presenta una maggior robustezza al rumore e quindi tende a convergere in modo più rapido. Ultimo algoritmo di ottimizzazione che si presenterà è Adam (Adaptive moment estimation) che fa parte dei sistemi di Adaptive Learning Rate. Questa tipologia di metodi di ottimizzazione nasce per rendere il learning rate α, che è l'iperparametro che maggiormente influisce sull'aggiornamento dei pesi, decrescente nel tempo. In questo modo si avrà un α alto all'inizio dell'apprendimento e quindi si raggiungerà velocemente la zona di minimo globale, poi con il passare delle epoche, si abbassa per avere una maggior sensibilità di aggiornamento. Adam è ad oggi, il metodo più efficiente per addestrare le reti neurali e quindi ci si soffermerà esclusivamente su questo tra i sistemi di Adaptive Learning Rate (Fig 2.12).

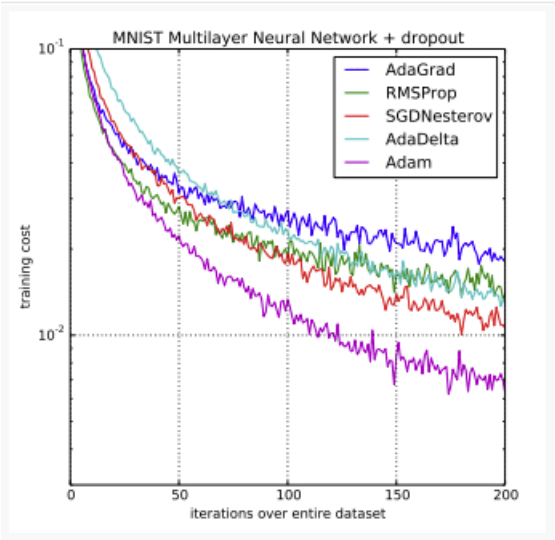


Figura 2.12: Prestazioni di diversi algoritmi di ottimizzazione a confronto in fase di training con l'utilizzo del dataset MNIST

L'algoritmo Adam riesce a combinare il concetto del momento con quello del learning rate adattivo. Adam utilizza sia il momento di ordine 1 che quello di ordine 2.

$$v_t = \beta_1 v_{t-1} + (1 - \beta_1)\nabla_w L$$
$$m_t = \beta_2 m_{t-1} + (1 - \beta_2)(\nabla_w L)^2$$

$$w_{new} = w_{old} - \alpha \frac{m_t}{\sqrt{v_t} + e}$$

Si può notare come abbiamo 4 iperparametri da scegliere β_1, β_2, α e e. Dalle prove sperimentali si sono determinati i valori di questi iperparametri:

1. α: Come negli altri algoritmi, è il learning rate e va scelto inizialmente, poi inseguito decrescerà. I valori più usati sono 10^{-1}, 10^{-2} o 10^{-3}.

2. β_1: Questo è il exponential decay rate per il primo momento che in genere è si imposta a 0.9.

3. β_2: Questo è il exponential decay rate per il secondo momento che in genere è si imposta a 0.999.

4. e: Questo coefficiente serve per non avere mai l'annullamento del denominatore della frazione $\frac{m_t}{\sqrt{v_t}+e}$. Ecco perché si usa un numero molto piccolo ed in genere è impostato a 10^{-8}.

Si è conclusa la parte generale sulle reti neurali e sui procedimenti di apprendimento che oggi sono maggiormente utilizzati. Nel capitolo successivo si parlerà del processamento delle immagini e delle tecniche maggiormente utilizzate oggi. Si partirà dalle informazioni di base, fino ad arrivare a delle tecniche di analisi con particolari reti neurali chiamate Convolutional Neural Network (CNN), che promettono grandi risultati a livello di classificazione e riconoscimento di anomalie in ambito industriale e non solo.

Capitolo 3

Metodi di elaborazione delle immagini

In ambito di controlli industriali sono spesso utilizzate tecniche di ispezione visiva. Due sono i metodi: ispezione manuale, fatta dall'operatore in linea anche con l'ausilio di una specifica strumentazione, o mediante sistemi automatizzati di visione (nel Capito X sarà riportato un esempio pratico di queste ispezioni). Quest'ultimi si compongono di un illuminatore, di un sensore ottico e di un obiettivo.

L'illuminatore gioca un ruolo molto importante nella determinazione del successo del controllo di visione in quanto va ad accentuare le caratteristiche del target (elemento soggetto al controllo) utili all'analisi mentre cancella quelle di disturbo. Un tipo di illuminazione può essere adatto per un determinato controllo ma non per un altro. L'illuminazione back light o retroilluminazione è perfetta per effettuare misurazioni del contorno dell'oggetto ispezionato, ma non è adatto a visualizzare imperfezioni superficiali in quanto l'informazione su quest'ultima viene completamente persa (Fig 3.1.b). Nel caso di difettosità superficiali si preferisce utilizzare un'illuminazione dark field che permette di accentuare eventuali anomalie. (Fig 3.1.c)

A seguire si ha il sensore ottico e l'obiettivo. Il sensore ottico (CMOS) è composto da pixels che trasformano l'impulso luminoso in un segnale digitale e quindi

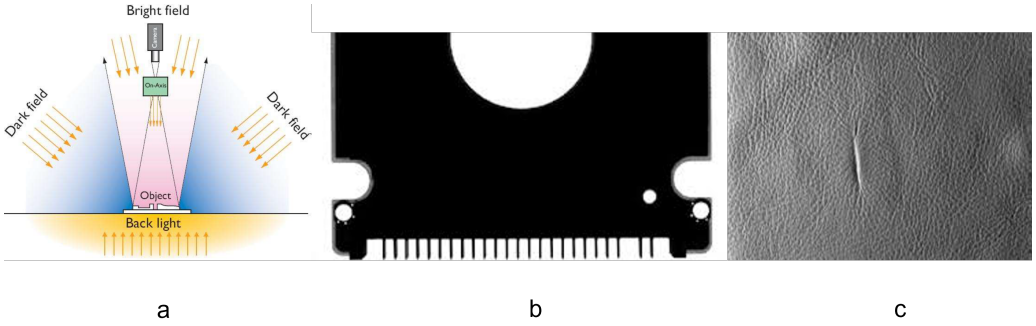


Figura 3.1: a) Principali configurazioni di illuminazioni b) Esempio di immagine con illuminazione back light c) Esempio di immagine con illuminazione dark field

di facile lettura per un elaboratore, mentre l'obiettivo ha la funzione di far arrivare nel sensore ottico un'immagine con maggior informazione possibile, ossia correttamente a fuoco e con il target di dimensioni adeguate rispetto all'immagine catturata.
Infine, il risultato è la conversione di una scena reale in un'immagine digitalizzata che verrà elaborata con tecniche di image processing per estrapolare le informazioni necessarie al fine del controllo. In genere gli obiettivi della machine vision sono l'Edge Detection, ossia usare tecniche per identificare i bordi e in seguito misurarli, Pattern Matching, che permette di identificare parti di oggetti mediante la correlazione con un'immagine modello, ed infine OCR, ossia la capacità di riconoscere i caratteri. Ultimamente, si parla sempre di più di computer vision in quanto con l'evoluzione degli algoritmi di intelligenza artificiale si riesce ad ottenere controlli di visione molto performanti e flessibili rispetto a quelli fatti con tecniche di machine vision.

3.1 Proprietà delle immagini digitali

Prima di introdurre i metodi di elaborazione delle immagini, si vedrà che cosa è un'immagine digitalizzata. Un'immagine digitale, F, è una matrice (o un insieme di esse) di dimensione mxn.

$$F = \begin{pmatrix} f(1,1) & \cdots & f(1,n) \\ \vdots & \ddots & \vdots \\ f(m,1) & \cdots & f(m,n) \end{pmatrix}$$

Ogni elemento di questa matrice viene chiamato pixel e si identifica come $f(i,j)$ dove $\forall i \in [1,m], \forall j \in [1,n], 0 \leq f(i,j) \leq W$. Quindi ogni pixel ha un valore intero compreso tra 0 e un valore W, che verrà spiegato in seguito. Le immagini digitali possiedono 3 caratteristiche principali:

1. **Risoluzione:** è la quantità di pixels che si usano per codificare l'immagine. Maggiori sono i pixels (mxn) maggiore sarà la qualità nella rappresentare di dettagli nell'immagine, ma anche la sua dimensione in bytes.

2. **Definizione:** indica la discretizzazione dell'intervallo dei valori di intensità che può assumere ogni singolo pixel ($f(i,j)$) e maggiore sono i livelli di discretizzazione maggiori saranno i bit che si dovranno dedicare ad ogni singolo pixel. Visto che sono i bit a determinare questi livelli, quest'ultimi saranno delle potenze di 2.

Unit8	8 bit	$2^8 = 256$	(nero=0,bianco=$256-1$)
Unit16	16 bit	$2^{16} = 65536$	(nero=0,bianco=$65536-1$)
Double	64 bit	$2^{64} = 1.8447x10^{19}$	(nero=0,bianco=$1.8447x10^{19}$)

   Una codifica a Unit8 è sufficiente per rendere alle immagini le corrette sfumature. Una codifica inferiore a 8 bit porta ad una netta divisione delle zone a diversa intensità (vedi Fig 3.2)

3. **Numero di canali:** Sono il numero di matrici che compongono l'immagine e queste matrici si chiamano canali o piani. Ogni canale rappresenta una particolare caratteristica dell'immagine e più canali ci sono e più l'immagine digitale è complessa.
   Quando l'immagine è composta solamente da un canale si parla di grayscale

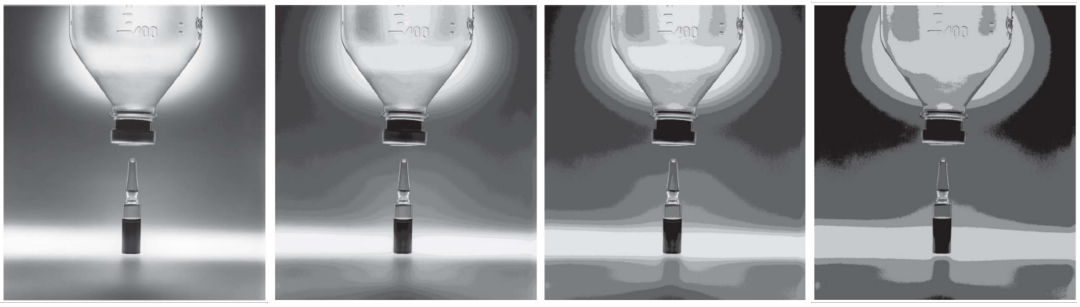


Figura 3.2: Esempio di diverse definizioni. I possibili livelli di intensità delle immagini da sx verso dx: 255,16,8 e 4

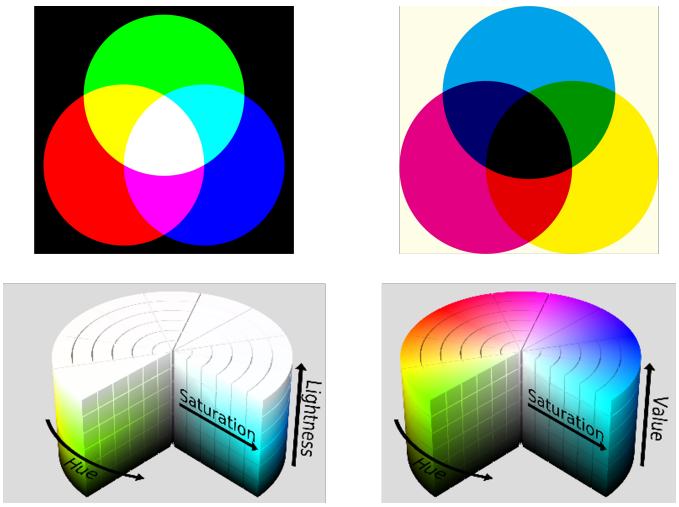


Figura 3.3: In alto a sx: RGB. In alto a dx: YMC. In basso a sx: HSL. In basso a dx: HSV

o di immagine pseudo-color, mentre si parla di immagine true-color quando si compone di 3 canali. Le tecniche di codifica per questa tipologia di immagine sono molteplici e ognuna con delle sue particolarità. La più utilizzata in ambito digitale è la codifica RGB (Red, Green, Blue), che sfrutta la composizione di rosso verde e blu per creare la maggior parte dei colori del visibile. Esistono molte altre codifiche e le più note in ambito digitale sono HSL (hue, saturation, luminance) e HSV (hue, saturation, value),mentre in ambito tipografico è usata la tecnica sottrativa YMC (yellow, magenta, cyan)(vedi Fig 3.3).

Un'altra codifica è la LAB, che secondo la pubblicazione "ColorNet: Inve-

Color Space	Accuracy	Time
RGB	78.89	26 secs
HSV	78.57	26 secs
YUV	78.89	26 secs
LAB	**80.43**	26 secs
YIQ	78.79	26 secs
XYZ	78.72	26 secs
YPbPr	78.78	26 secs
YCbCr	78.81	26 secs
HED	78.98	26 secs
LCH	78.82	26 secs

Figura 3.4: Risultati ottenuti con CNN con differenti codifiche a 3 canali

stigating the importance of color spaces for image classification" pubblicata nel 2019 da Shreyank N Gowda e Chun Yuan è maggiormente prestante per operazioni di classificazione mediante reti neurali convoluzionali (vedi Fig 3.4). Non si approfondirà l'argomento della colorimetria in quanto non è lo scopo della trattazione.

Un'immagine digitale nella sua forma più generale sarà indicata nel seguente modo:

$$\text{Immagine digitale} = (m, n, c) = [F_1, F_2, \ldots, F_c]$$

Dove m sta per il numero di pixels verticali (numero di righe) delle matrici, n il numero dei pixels orizzontali (numero delle colonne) e c il numero di canali o piani. Nelle immagini multicanale, la dimensione delle matrici di ogni canale sarà la stessa.

$$dim(F_1) = dim(F_2) = \ \ldots \ = dim(F_c)$$

Portando un esempio concreto, quando si parla di un'immagine 256x256 monocromatica a 8 bit, si ha un singolo canale ossia una singola matrice di valori dei pixels, il valore assoluto dei pixels ($f(i, j)$) variano tra 0 e 255 ($2^8 = 256$ valori possibili) e ho una risoluzione pari a mxn ossia 256x256.

$$\text{Gray scale} = (256, 256, 1) = F$$

$$W = 255$$

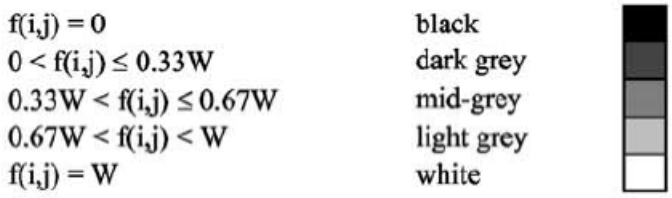


Figura 3.5: Correlazione tra la gradazione di grigio con l'intensità di ogni pixel

Nel caso si avesse un'immagine RGB, avremo 3 canali ma riguardo al concetto di definizione e di risoluzione non cambia nulla rispetto all'immagine monocromatica.

$$\text{Immagine RGB} = (m, n, 3) = [F_1, F_2, F_3] = [R, G, B]$$

$$R = \ r(i,j) \quad G = g(i,j) \quad B = b(i,j)$$

$$W = 255$$

3.2 Tecniche base di elaborazioni delle immagini

In questo paragrafo si spiegheranno le tecniche di base propedeutiche per una corretta comprensione del modello di computer vision sviluppato nel progetto.

3.2.1 Point-Point Operation

Le prime operazioni che vedremo sono quelle Point-Point Operation con singola immagine, e poi con due o più. Nelle operazioni a singola immagine, ogni pixel originale ($a(i,j)$) viene trasformato mediante una funzione ($g(x)$) in un nuovo pixel ($c(i,j)$). A livello matematico, la relazione è la seguente:

$$c(i,j) = g(a(i,j)) \quad \forall i \in [1,m], \forall j \in [1,n]$$

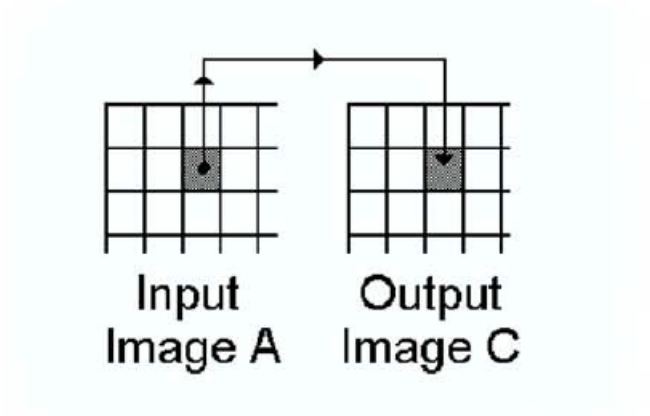


Figura 3.6: Rappresentazione grafica Point-Point Operation a singola immagine

Nella Fig 3.7, sono riportate le principali operazioni Point-Point a singola immagine che si usano attualmente.

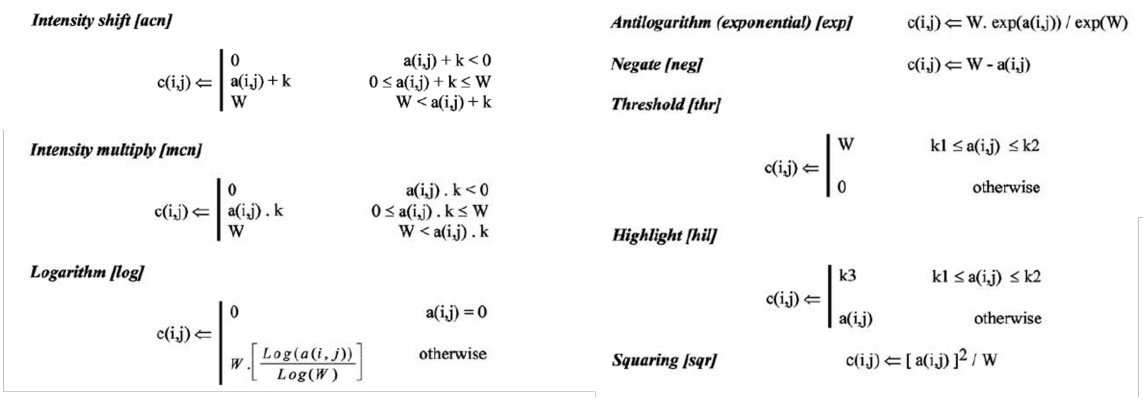


Figura 3.7: Principali operazioni Point-Point a singola immagine

Di particolare importanza è l'operazione Threshold, in quanto trasforma un'immagine monocromatica in un'immagine binaria (bianco e nero). E' un metodo molto utilizzato in ambito del controllo industriale ed inoltre questo formato permette di effettuare le operazioni morfologiche (erosione e dilatazione).
Le operazione Point-Point con due o più immagini si differenziano dalle operazioni precedenti solo per la funzione $g()$, che in questo caso non sarà univariata ma multivariata, ossia prende in input lo stesso pixel ma da più immagini. Date delle coordinate spaziali (i, j), l'equazione della relazione è la seguente:

$$c(i,j) = g(a(i,j), b(i,j)) \quad \forall i \in [1,m], \forall j \in [1,n]$$

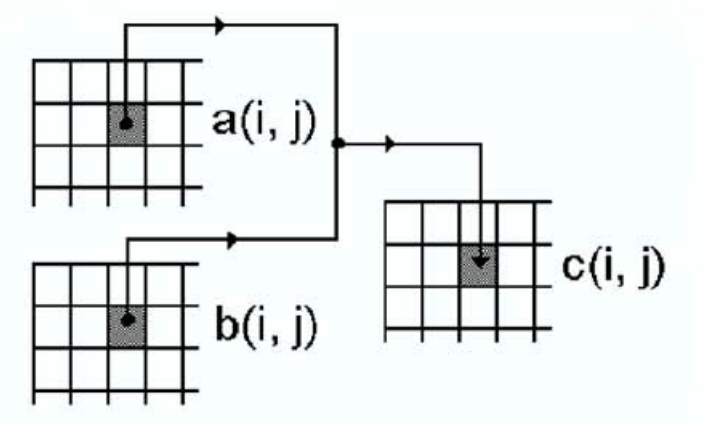


Figura 3.8: Rappresentazione grafica Point-Point Operation con due immagini

Nella Fig 3.9, sono riportate le principali operazioni Point by Point con due o più immagini.

Add [add]	$c(i,j) \Leftarrow [\ a(i,j) + b(i,j)\ ] / 2.$
Subtract [sub]	$c(i,j) \Leftarrow [\ (\ a(i,j) - b(i,j)\ ) + W\ ] / 2$
Multiply [mul]	$c(i,j) \Leftarrow [\ a(i,j).b(i,j)\ ] / W$
Maximum [max]	$c(i,j) \Leftarrow MAX\ [\ a(i,j), b(i,j)\ ]$
Minimum [min]	$c(i,j) \Leftarrow MIN\ [\ a(i,j), b(i,j)\ ]$

Figura 3.9: Principali operazioni Point-Point con due immagini

3.2.2 Local Operators

Un'altra tipologia di operazione è data dagli Local Operators che usano il concetto di neighbourhood. In queste operazioni, dato un pixel $f(i, j)$, si considerano anche i suoi vicini, ossia quelli che si trovano nel suo intorno. L'estensione del l'intorno è dato dalle dimensioni del filtro (chiamato anche kernel o maschera), che in genere è quadrato con dimensioni 3x3, ma si usano anche filtri 5x5 e 7x7. In questa trattazione prenderemo come riferimento il filtro 3x3. In Fig 3.10 è presentata la convenzione che si userà nella trattazione per indicare i vicini di un particolare pixel dell'immagine originale. Nella matrice di sinistra si ha la notazione esplicita dove si utilizzano le coordinate del pixel di riferimento per identificare i suoi vicini.

Nella matrice di destra invece si è utilizza una nomenclatura per identificare ogni pixel (dalla A alla I).

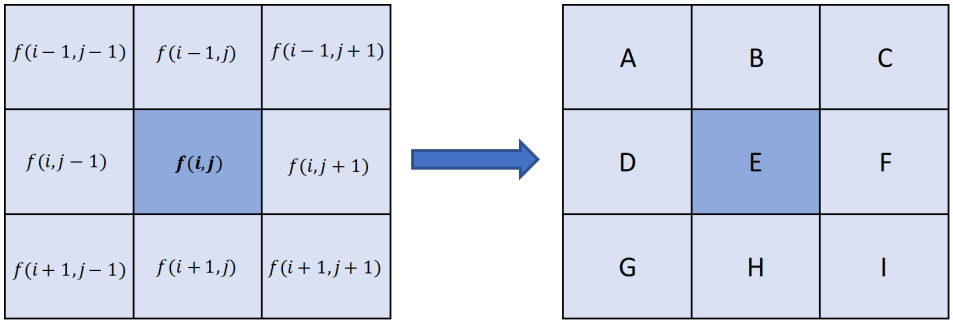


Figura 3.10: Convenzione per rappresentare il concetto di neighbourhood

L'equazione matematica che sussiste è la seguente:

$$c(i,j) = g(a(i-1,j-1), a(i-1,j), a(i-1,j+1), a(i,j-1), a(i,j), a(i,j+1), a(i+1,j-1), a(i+1,j), a(i+1,j+1))$$

$$\forall i \in [1,m], \forall j \in [1,n]$$

$$c(i,j) = g(A,B,C,D,E,F,G,H,I)$$

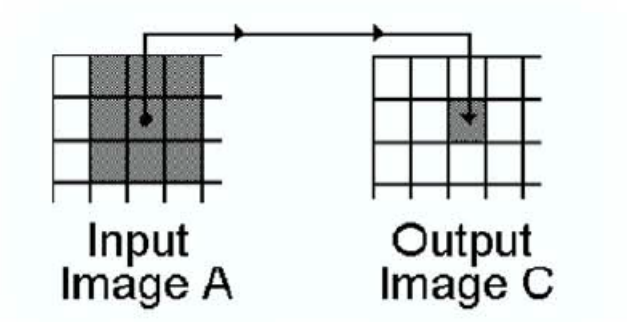


Figura 3.11: Rappresentazione grafica Local Operators con finestra 3x3

Andando sul dettaglio, ci possono essere 3 tipologie di neighbourhood (Fig 3.12):

1. 4-neighbors (N_4), che vengono chiamati anche edge neighbors

$$c(i,j) = g(B,D,E,F,H)$$

2. D-neighbors (N_D), che vengono chiamati anche point neighbors

$$c\,(i,j) = g\,(A,C,E,G,I)$$

3. 8-neighbors (N_8), che è l'unione delle prime due tipologie

$$c\,(i,j) = g\,(A,B,C,D,E,F,G,H,I)$$

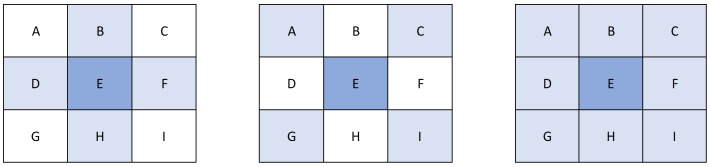


Figura 3.12: Le tre tipologie di neighbourhood. Partendo da sinistra si ha la 4-neighbors, D-neighbors e 8-neighbors

I Local Operators posso essere di due tipologie lineari o non lineari, su cui non ci soffermeremo. Tra i lineari le operazioni maggiormente utilizzate sono le convoluzioni ossia il prodotto scalare tra i coefficienti del kernel che sono chiamati pesi e si indicano con w e i relativi pixel dell'immagine. A livello matematico si scrive nel seguente modo:

$$c\,(i,j) = (w * \ a)(i,j) = \sum_{s=-\alpha}^{\alpha} \sum_{t=-\beta}^{\beta} w(s,t)a(i-s,j-t)$$

Dove $\alpha = \frac{(N-1)}{2}$,$\beta = \frac{(M-1)}{2}$ e M e N sono rispettivamente le due dimensioni del filtro. Molto spesso il filtro utilizzato è quadrato, quindi $M = N$ e $\alpha = \beta$. L'equazione precedente assumerà la seguente forma:

$$c\,(i,j) = (w * \ a)(i,j) = \sum_{s=-\alpha}^{\alpha} \sum_{t=-\alpha}^{\alpha} w(s,t)a(i-s,j-t)$$

Attualmente esistono molti filtri che utilizzano la convoluzione per apportare modifiche o accentuare dei particolari nelle immagini. Nella figura 3.13 si riportano alcuni Smoothing filters (Filtro gaussiano o filtro passa-basso, filtro mediano circolare, filtro mediano rettangolare) e nella figura 3.14 si riportano i due più famosi filtri di Edge Detection Filters, ossia Sobel operator e Laplace operator. Laplace operator nella realtà è un filtro passa-alto, quindi riesce ad identificare bene

tutte le variazioni repentine nell'immagine ed ecco perchè si utilizza per l'edge detection (lo svantaggio è che ad alta frequenza si trova anche il rumore)

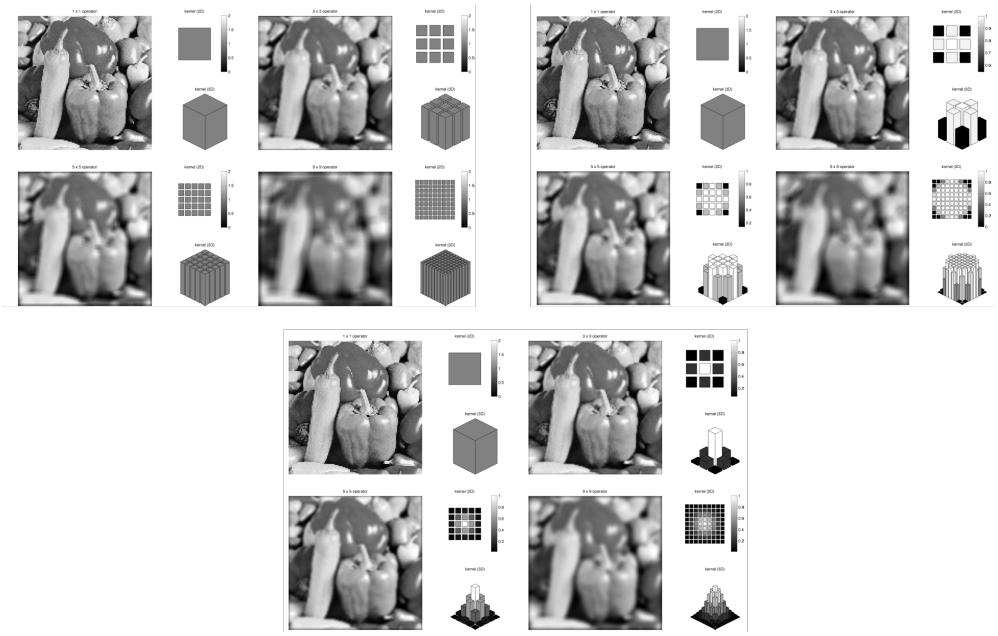


Figura 3.13: In alto a sinistra si hanno i risultati di quattro convoluzioni ottenute con filtri mediani rettangolari di varie dimensioni. In alto a destra si hanno i risultati di quattro convoluzioni ottenute con filtri mediani circolari di varie dimensioni. In basso si hanno i risultati di quattro convoluzioni ottenute con filtri mediani gaussiani di varie dimensioni

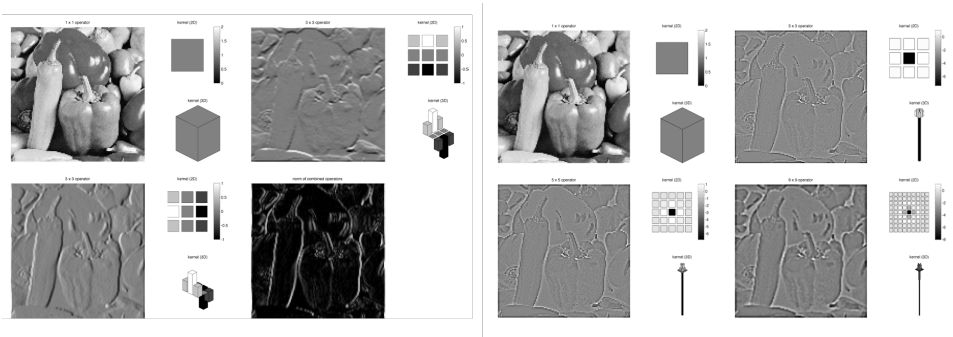


Figura 3.14: A sinistra si hanno i risultati di quattro convoluzioni ottenute due filtri di Sobel, uno per identificare i bordi verticali e uno per identificare quelli orizzontali. A destra si hanno i risultati di quattro convoluzioni ottenute con filtri di Laplace di varie dimensioni.

3.3 Utilizzo delle reti neurali per elaborazione delle immagini

Negli ultimi anni, la ricerca sulla computer vision ha portato allo sviluppo di particolari reti neurali che usano la convoluzione per ottimizzare l'analisi delle immagini. Rispetto alle tecniche di machine vision tradizionali, la computer vision con CNN ha i seguenti vantaggi:

1. Più oggettivo, in quanto nella machine vision l'efficacia di controllo dipende dall'esperenzienza e competenze del tecnico addetto, che seglie la sequenza e la tipologia di elaborazione d'immagine da fare. Diversamente, la computer vision usando tecniche di intelligenza artificiale riesce a costruirsi un modello per questa analisi senza l'ausilio umano.

2. Più flessibile. Nella machine vision in genere per ogni tipologia di prodotto è predisposta ua ricetta specifica, in quanto cambiamenti minimi di colore, posizione o componenti posso influenzare pesantemente il risultato del controllo. Con la computer vision è possibile far apprendere al sistema le features di base delle immagine, ossia dove è presente l'informazione principale per il controllo e questo permette al sistema di funzionare anche con prodotti delle stesse famiglia senza dover cambiare ricetta.

Le reti neurali convoluzionali (CNN) sono utilizzate per 3 ambiti diversi:

1. **Classificazione:** sono le reti che sono maggiormente utilizzate. Queste reti, data una immagine in input sono in grado di restituire la probabilità di appartenenza ad ogni singola classe (tra quelle usante nella fase di training). Le reti maggiormente note in letteratura che fanno questo sono VGG-16, ResNet50, Inception, Efficientnet.

2. **Object detection:** sono delle reti che oltre ad effettuare una classificazione riescono anche ad identificare spazialmente l'oggetto nell'immagine. In genere l'oggetto della ricerca è evidenziato mediante un boundary box.

L'algoritmo maggiormente diffuso che esegue object detection si chiama YOLO.

3. **Riconoscimento delle anomalie:** Sono delle reti che nella fase di training sono allenate esclusivamente con immagini di una classe. Questo permette all'algoritmo di identificare tutte le immagine che hanno qualcosa di diverso dalle immagine date in allenamento, quindi delle anomalie. In genere questa tipologia di rete è utilizzata quando non è possibile conoscere a priori le classi o quando si hanno classi molto sbilanciate, ossia poche immagine di una classe e molte dell'altra. Attualmente il sistema maggiormente utilizzato per riconoscere anomalie nelle immagini è il Convolutional Autoencoder (CAE). In seguito per lo studio di fattibilità, si utilizzerà questa tipologia di algoritmo, per cui si spiegherà meglio il suo funzionamento in seguito.

La logica che c'è dietro le reti neurali convoluzionali è la stessa delle reti neurali trattate nel capitolo precedente. Quindi il concentto di layer, delle funzioni di attivazione, della bachpropagation e degli algoritmi di ottimizzazione per la diminuzione della funzione di perdita rimane invariato. L'unica cosa che varia rispetto alle precedenti reti neurali è la struttura del layer, non si avranno più dei percettroni ma dei filtri i quali elementi sono i pesi che si andranno a determinare durante la fase di addestramento.

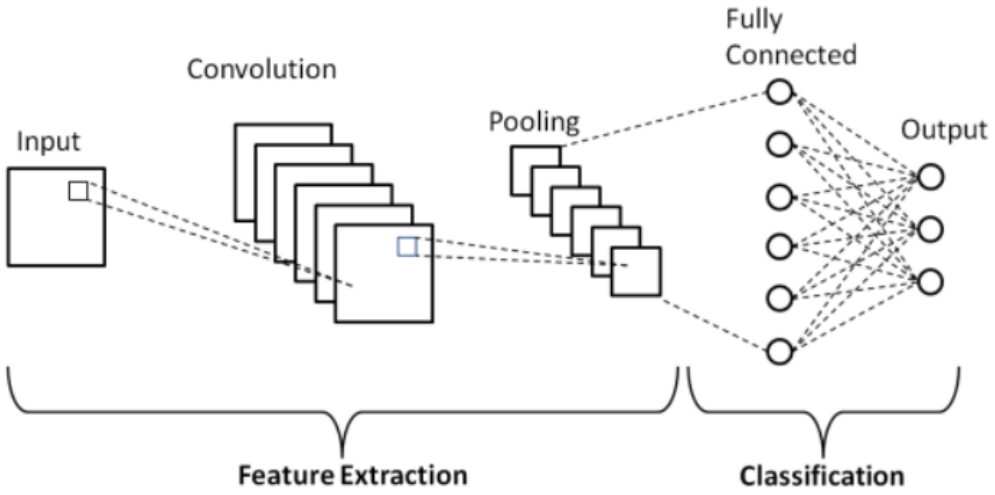


Figura 3.15: Struttura base delle Convolutional Neural Network

In figura 3.15 si presenta un semplice esempio di rete neurale convoluzionale per spiegare la struttura di base. Le CNN sono composte da due parti generalmente, la prima parte è chiamata di feature extraction, dove si hanno i layers di convoluzioni e di pooling, la seconda parte è chiamata di classification o anche fully connection ed è una classica rete neurale a percettroni. La parte di feature extraction si chiama così in quanto è la parte che serve ad estrarre le informazioni più importanti dall'immagine. Mendiante la convoluzione tra l'immagine di input e i filtri, i quali pesi vengono aggiornati e ottimizzati nella fase di addestramento, si ha in output un'immagine con le stesse dimensioni ma con i canali pari al numero di filtri. La fase successiva è la fase di pooling che è un'operazione locale non lineare che generalmente mediante un filtro estrae da ogni canale i pixels a maggior valore (operazione di Max Pooling). Quindi il risultato di una convoluzione e di un pooling è un'immagine di dimensioni ridotte ma con maggior canali. Iterando questo processo si riesce ad estrarre tutte le informazioni necessarie per poi utilizzare la seconda parte della CNN per effettuare una classificazione. L'uso di CNN al posto delle classiche reti neurali per l'elaborazione delle immagini portano una grande diminuzione del costo computazione senza penalizzare le prestazioni.

3.4 Metriche di valutazione della qualità delle immagini

Durante la trasmissione e compressione dell'immagine, quest'ultima potrebbe subire delle distorsioni geometriche o l'introduzione di rumore che potrebbero compromettere il risultato finale dell'elaborazione. Un esempio nell'ambito della edge detection, una leggera distorsione dell'immagine potrebbe portare ad un aumento dell'errore medio di misura delle quote del oggetto da indagare, aumentando la probabilità di avere pezzi fuori tolleranza e quindi aumentare i falsi scarto. Ecco perchè molto importante è la valutazione della qualità dell'immagine (IQA). Sono disponibili molte tecniche e metriche per la valutazione della qualità e si dividono in due grandi categorie:

1. **Approcci Full-Reference (FR):** gli approcci FR si concentrano sulla valutazione della qualità di un'immagine di prova rispetto ad un'immagine di riferimento. Questa immagine di riferimento è considerata come l'immagine di qualità perfetta che significa la verità di base.

2. **Approccio No-Reference (NR):** le metriche NR si concentrano solo sulla valutazione della qualità di un'immagine di prova. In questo metodo non viene utilizzata alcuna immagine di riferimento.

In questa trattazione si presenteranno solo le principali metriche Full-Reference, ossia MSE e SSIM.

3.4.1 Mean Square Error (MSE)

MSE è lo stimatore più comune della metrica di misurazione della qualità dell'immagine. Questa misura è la differenza quadratica media tra i valori dei pixel dell'immagine di riferimento, che si assume abbia qualità massima, e quelli dell'immagine campione. La fomula matematica di riferimento è la seguente:

$$MSE = \frac{1}{MN} \sum_{n=0}^{M} \sum_{m=1}^{N} [\hat{f}(n,m) - f(n,m)]$$

Dove $f(n, m)$ è l'immagine di campione mentre $\hat{f}(n, m)$ è l'immagine di riferimento.
MSE è estremamente semplice da implementare, ma è una metrica molto sensibile a piccole traslazioni o rotazione dell'immagine. Questo è dovuto proprio alla semplicità della formula che opera pixel to pixel senza tenere conto dell'intorno dell'immagine. Se MSE è uguale a 0 significa che l'immagine campione è perfettamente uguale all'immagine di riferimento, al crescere del MSE, generalmente, corrisponde ad una maggiore diversità o diminuzione di qualità.

3.4.2 Structure Similarity Index Method (SSIM)

La metrica SSIM viene calcolata in base a tre aspetti principali definiti come luminanza, contrasto e termine strutturale o di correlazione. Questo indice è una combinazione di moltiplicazione di questi tre aspetti:

$$SSIM(f, \hat{f}) = [l(f, \hat{f})]^{\alpha}[c(f, \hat{f})]^{\beta}[s(f, \hat{f})]^{\gamma}$$

Qui, l è la luminanza (usata per confrontare la luminosità tra due immagini), c è il contrasto (usato per differenziare gli intervalli tra la regione più luminosa e più scura di due immagini) e s è la struttura (usata per confrontare il modello di luminanza locale tra due immagini per trovare la somiglianza e la dissomiglianza delle immagini) e α, β e γ sono le costanti positive.
Anche in questo caso la luminanza, il contrasto e la struttura di un'immagine possono essere espressi separatamente come:

$$l(f, \hat{f}) = \frac{2\mu_f\mu_{\hat{f}} + C_1}{\mu_f^2 + \mu_{\hat{f}}^2 + C_1}$$
$$c(f, \hat{f}) = \frac{2\sigma_f\sigma_{\hat{f}} + C_2}{\sigma_f^2 + \sigma_{\hat{f}}^2 + C_2}$$
$$s(f, \hat{f}) = \frac{\sigma_{f\hat{f}} + C_3}{\sigma_f\sigma_{\hat{f}} + C_3}$$

dove μ_f e $\mu_{\hat{f}}$ sono le medie locali, σ_f e $\sigma_{\hat{f}}$ sono le deviazioni standard e $\sigma_{f\hat{f}}$ è la covarianza incrociata delle immagini f e $\hat{f}$. Se $\alpha = \beta = \gamma = 1$ la formula del SSIM si semplifica e diventa:

$$SSIM(f, \hat{f}) = \frac{(2\mu_f \mu_{\hat{f}} + C_1)(2\sigma_{f\hat{f}} + C_2)}{(\mu_f^2 + \mu_{\hat{f}}^2 + C_1)(\sigma_f^2 + \sigma_{\hat{f}}^2 + C_2)}$$

Questa metrica varia da 0 ad 1. Quando SSIM è 1 significa che l'immagine campione è uguale all'immagine di riferimento, più il valore si abbassa più si ha una perdita di qualità o di somiglianza rispetto a quella di riferimento.
Questa metrica, rispetto al MSE è più costosa a livello computazione ma riesce a percepire cambiamenti strutturali dell'immagine, in quanto usa grandezze globali (media, varianza e covarianza) rispetto a grandezze locali (differenza pixel-pixel).

Capitolo 4

Valutazioni sui controlli di qualità

Prima di passare alla descrizione del progetto nella sua interezza, è di fondamentale importanza dare delle nozioni generali sulla diagnostica industriale e su quali metriche misurare l'efficacia di un sistema di controllo qualità. La diagnostica è la procedura per la determinazione dello stato di salute di un oggetto o di un suo componente e comprende 3 fasi:

1. **Misura:** Rilevazione delle caratteristiche che portano l'informazione sullo stato di salute.

2. **Classificazione:** Classificare mediante le caratteristiche rilevate i componenti in due o più insiemi, per esempio sano e difettato

3. **Decisione:** Decidere se accettare o scartare l'oggetto o il componente.

Queste 3 fasi sono anche la base per la progettazione di un sistema di controllo qualità. Si può intuitivamente capire, che all'aumentare delle caratteriste identificate nella fase di misura e dalla difficoltà nel misurarle, aumenterà la complessità della fase di classificazione, in quanto non sarà sufficiente una relazione lineare a separare correttamente i componenti difettati da quelli sani. In altre parole un sistema di controllo è tanto più efficace quanto più riesce a separare le distribuzioni delle diverse classi.

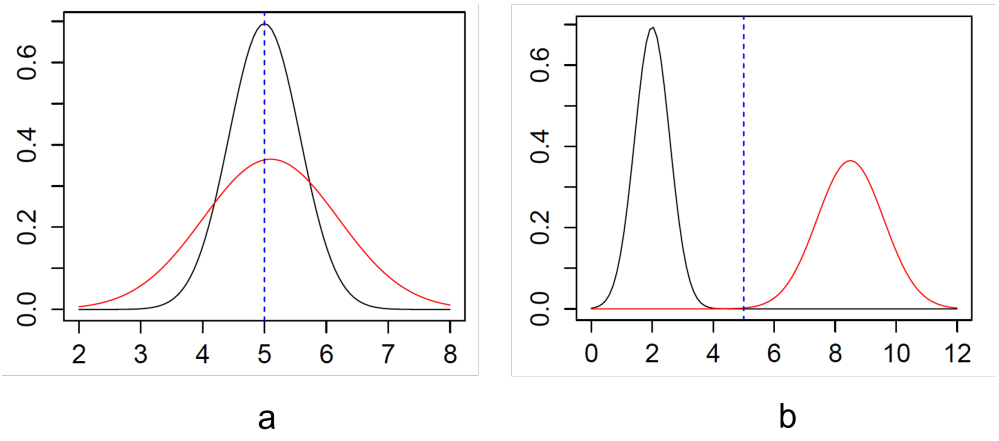


Figura 4.1: a) Caso in cui la media della distribuzione dei componenti difettati coincide con la media dei valori dei componenti sani. b) Caso in cui le due classi sono completamente separate

Nella figura 4.1si può notare i due casi estremi, da una parte un sistema di controllo non utile in quanto la curva rossa (difetti) ha il valor medio coincidente con il valor medio della curva nera (sani). Questo sta a significare che le due classi sono completamente sovrapposte ed impossibile da dividere. L'altra figura invece rappresenta il caso in cui il sistema di controllo riesce a dividere perfettamente le due classi infatti le due distribuzioni non si intersecano. Il caso più comune è rappresentato in figura 4.2, ossia l'intersezione avviene alle estremità (code) delle distribuzioni.

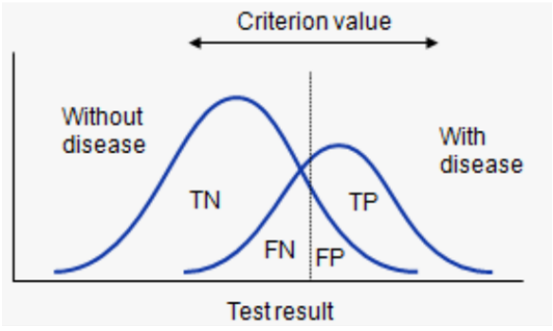


Figura 4.2: Distribuzioni generali di due classi

In questi casi va scelta una soglia x_d che viene chiamata anche criterion value o cut-off che determina 4 zone nelle due distribuzioni:

1. **True Positive (TP):** è la zona della distribuzione probabilistica dei componenti sani che sono realmente codificati sani dal sistema di controllo

2. **True Negative (TN):** è la zona della distribuzione probabilistica dei componenti difettati che sono realmente codificati difettati dal sistema di controllo

3. **False Positive (FP):** è la zona della distribuzione probabilistica dei componenti difettati che il sistema di controllo reputa sani.

4. **False Negative (FN):** è la zona della distribuzione probabilistica dei componenti sani che il sistema di controllo reputa difettati.

Si capisce che quando si hanno sia FN=0 che FP=0 si ritorna al caso ideale delle due distribuzioni completamente divise. Spesso in ambito pratico queste aree di probabilità diventano numeri concreti ed ecco perchè si usa un'altro strumento per visualizzare questi risultati che si chiama Confusion Matrix (vedi figura 4.3).

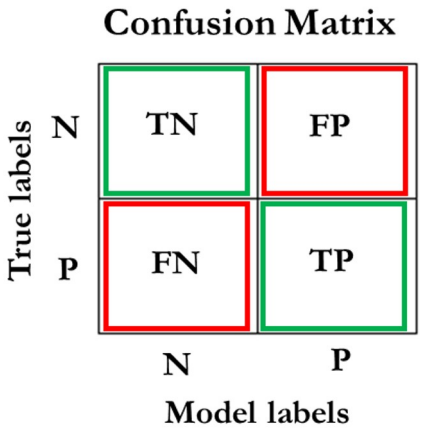


Figura 4.3: Confusion Matrix

Sulla diagonale principale si trovano i valori assoluti o percentuali di TN e TP, mentre sulla diagonale secondaria si trovano i valori assoluti o percentuali degli errori che commette il controllo di qualità cioè FN e FP. Inoltre la sommatoria dei valori delle righe corrisponde ai valori delle singole classi (difetti e sani), mentre la sommatoria dei valori delle colonne rappresentano i valori degli scarti

e dei componenti sani determinati dal sistema di controllo. Da questi valori (TP, TN, FP, FN) si possono ricavare delle metriche per misurare l'efficacia del controllo qualità. Le maggiormente utilizzate sono il recall, precision e la loro media armonica F1-score.

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

$$\text{F1-score} = \frac{2 * precision * recall}{precision + recall}$$

Tutte queste 3 metriche variano tra 1 e 0 e si nota come all'aumentare dei FP la precision diminuisce mentre all'aumentare dei FN diminuisce il recall. In ambito produttivo se la precision è troppo bassa significa che il controllo non identifica bene i difetti, questo porta ad un aumento delle non conformità nel processo produttivo. Se invece il recall è basso significa che il controllo assegna la non conformità a molti componenti che nella realtà sono conformi, abbassando il parametro di efficienza del processo produttivo. F1-score riesce ad aggregare queste due metriche ed ecco perchè è molto utilizzato per effettuare valutazioni e confronti tra vari controlli di qualità.

Precedentemente si è introdotto il concetto di soglia x_d per determinare le aree delle distribuzioni che corrispondono a TP, TN, FP e FN. Il valore di soglia è un parametro che viene scelto in base a considerazioni economiche e dalle politiche aziendali.

Quando si hanno problemi a classi sbilanciate le metriche precision recall e F1-score, così presentante, non riescono a dare la giusta sensibilità. In questi casi è possibile utilizzare le macro metriche ossia, macro F1-score, macro Recall e macro Precision, che essendo le medie delle rispettive metriche di ogni classe, permettono di dare la stessa importanza alle classi, senza considerare il numero assuluto di elementi che la compongono.Quindi:

$$macro\ precision = \frac{precision_{OK} + precision_{NOK}}{2}$$

$$macro\ recall = \frac{recall_{OK} + recall_{NOK}}{2}$$

$$\text{macro F1-score} = \frac{\text{F1-score}_{OK} + \text{F1-score}_{NOK}}{2}$$

Dove per 2 è il numero delle classi che saranno prese in considerazione, in questo caso OK e NOK. Per problemi multi classe il denominatore sarà pari al numero delle classi.

Quest'ultime saranno le metriche che si utilizzeranno per la valutazione dei sistemi di controllo nei capitoli seguenti proprio per il problema dello sbilanciamento, in quanto la classe degli scarti di saldatura rappresentano in genere all'incirca 1% della produzione.

Capitolo 5

Analisi preliminare della linea

5.1 Layout linea automatica

Il progetto di sperimentazione del sistema di controllo saldature mediante computer vision è stato condotto su una linea di assemblaggio automatica a layout circolare il quale scopo è quello di effettuare 4 saldature stagno per collegare due microinterruttori al carter che andrà a comporre la struttura portante di una famiglia di serrature baule per diversi modelli di automobili (vedi figura 5.1).

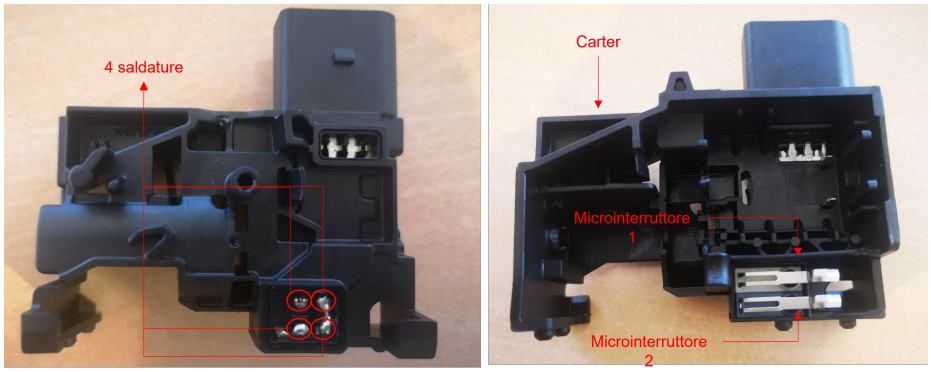


Figura 5.1: A sinistra sono evidenziate le 4 saldature effettuate tra il cater e i due microinterruttori. A destra, si evidenziano i due microinterruttori con la vista dal basso del carter

In dettaglio la linea si compone di 10 stazioni (vedi figura 5.2). La stazione ST00 è l'unica stazione dove è presente un operatore che esegue manualmente il

preassemblaggio tra il carter e i due microinterruttori. Nella stazione ST10 si ha il controllo visivo dei microinterruttori e si verifica la loro assenza/presenza e il loro colore (modelli diversi hanno microinterruttori con colori diversi). A seguire si hanno le 4 stazioni di saldatura automatiche (ST20, ST30, ST40, ST50) che eseguano il collegamento permanente tra i reofori dei microinterruttori e la piazzola dedicata sul carter. In seguito si hanno una stazione di tranciatura (ST60, non importante ai fini di questa trattazione), la stazione di controllo di qualità della saldatura effettuata con un sistema di machine vision della Omron (ST70), la stazione di collaudo elettrico più controllo visivo eseguito dall'operatore (ST80) mediante un microscopio industriale collegato ad un monitor presente nella stazione ST00 ed infine si ha la stazione di scarico (ST90) che porta il prodotto assemblato in un convogliatore a nastro motorizzato se gli esiti dei controlli in ST70 e ST80 danno la conformità. Se così non fosse, il robot in ST90 deposita l'assemblato nella cassetta degli scarti.

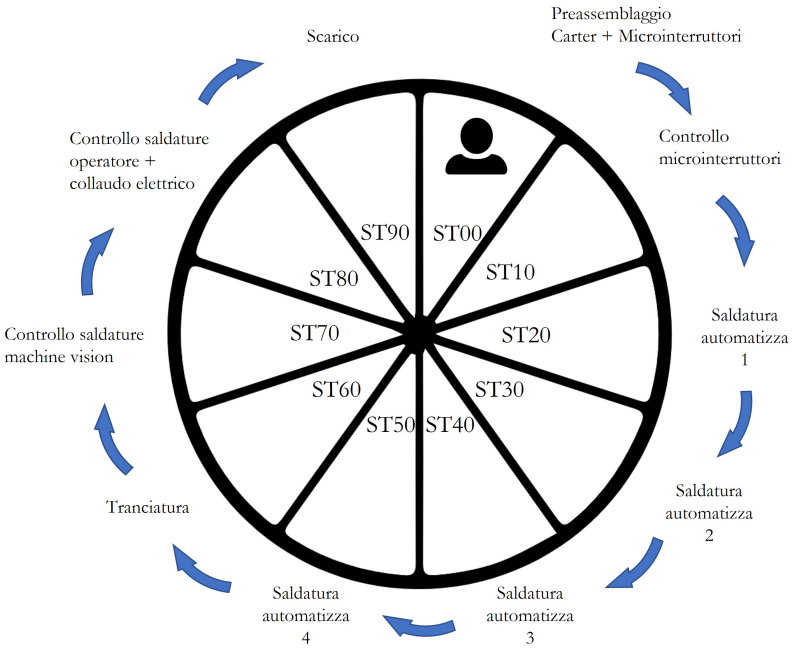


Figura 5.2: Layout schematico della linea analizzata

5.2 Introduzione alla tecnica e processo di saldatura

Prima di parlare dei sistemi di controllo effettuati sulla saldatura, si introduce una breve presentazione della tecnologia utilizzata e come è implementata nella linea di riferimento.
La tecnica di giunzione più comunemente utilizzata per gli assemblaggi elettronici è la brasatura dolce (soldering) ed è caratterizzata da un principio chimico/fisico chiamato bagnatura (wetting). Per comprendere tale fenomeno è sufficiente pensare ad una superficie metallica e a una goccia di lega brasante fusa che entrano in contatto e si otterrà due condizioni opposte, ovvero:

1. La lega saldante fusa, entrando in contatto con la superficie metallica, si deposita allargandosi uniformemente sulla stessa fino a riempirne tutti gli spazi: in questo caso la lega brasante ha bagnato la superficie (wetting condition- figura 5.3.a).

2. La lega saldante fusa, entrando in contatto con la superficie metallica, non si deposita allargandosi sulla stessa tendendo a mantenere la sua forma sferica iniziale: in questo caso la lega brasante non ha bagnato la superficie (no wetting condition- figura 5.3.b).

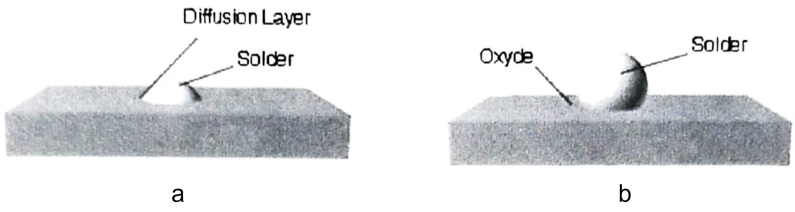


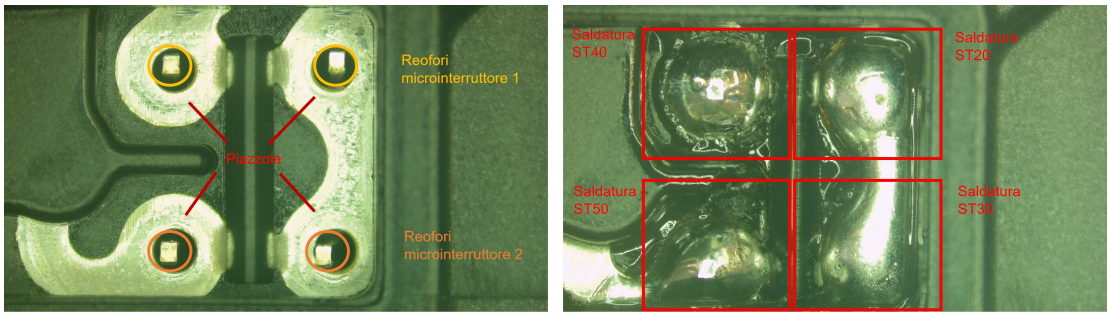
Figura 5.3: a) Wetting Condition. b) no-Wetting Condition

La condizione necessaria per ottenere una buona connessione saldata è quella in cui tutte le superfici da unire vengano completamente bagnate dalla lega brasante fusa, pertanto, per ottenere ciò è necessario considerare tutti quelli che sono i fattori che favoriscono (o non) una buona bagnatura. Questi fattori sono la natura

metallo da bagnare, lo stato superficiale, la tensione superficiale, le irregolarità superficiali e la formazione di composti intermetallici.
Importante inoltre, è la scelta della lega brasante che deve possedere elevata conducibilità termica, elevata conducibilità elettrica, buona bagnabilità delle superfici, basso punto di fusione, determinate caratteristiche meccaniche e costi contenuti. In genere la lega brasante utilizza maggiormente sono leghe di stagno senza piombo (Lead-free), a causa delle normative sulla sua tossicità. Le leghe lead-free quindi sono leghe di stagno combinato con altri elementi per aumentare certe caratteristiche elencate precedentemente e sono: Argento (Ag), Indio (In), Zinco (Zn), Antimonio (Sb), Rame (Cu), Bismuto (Bi), Cadmio (Cd).
La tecnica di saldatura che si utilizza nella linea nelle stazioni ST20, ST30, ST40, e ST50 è la brasatura dolce a filo robotizzata, che simula la saldatura a filo manuale fatta da un operatore. L'end-effector del robot è composto da un rocchetto di filo di lega brasante che viene srotolato a velocità costante da un attuatore nel momento della saldatura. Questo filo, mediante una guida viene indirizzato verso la punta di saldatura che per l'effetto Joule si trova a temperature tra 300 °C e 360°C, portando a fusione il filo. La lega brasante accumulata sulla punta per effetto della gravità fluisce verso il basso dove incontra il reoforo del microinterruttore e la piazzola del carter bagnandoli e quindi effettuando il giunto. In figura 5.4.c si possono vedere le varie fasi della saldatura.

Figura 5.4: a) Rocchetto di filo di lega brasante. b) Tipica configurazione per la punta di saldatura e il filo per effettuare il giunto. c) Gli steps per effettuare una corretta giunzione nella brasatura dolce robotizza a filo

Tornando al caso specifico della linea analizzata, in figura 5.5.a viene presentata la configurazione dei componenti prima della fase di saldatura. Come si può vedere si hanno 4 reofori con 3 piazzole diverse (una con fori) e quindi richiedono 4 giunzioni distinte. Quest'ultime sono effettuate dalle 4 stazioni di saldatura automatica, ognuna settata per una specifica giunzione ed ecco perchè in seguito le singole saldature saranno identificate con il nome della stazione (vedi figura5.5.b). Il risultato finale del processo di saldatura è visibile nella figura5.5.b dove tutte le giunzioni sono avvenute correttamente e senza difetti.



a b

Figura 5.5: a) Immagine del carter più microinterruttori prima della giunzione con lega brasante. b) Immagine del carter più microinterruttori dopo la saldatura.

Nella pratica, il processo descritto è soggetto a molte variabili che posso portare a giunti non conformi con gli standard richiesti dal cliente o dalla normativa. I difetti che maggiormente si verificano nel processo di brasatura dolce in questa linea sono:

1. Possibili cortocircuiti tra le saldature ST40 e ST20, o ST50 e ST30. Meno probabili sono i cortocircuiti tra ST40 e ST50, o ST40 e ST30, o ST50 e ST20.

2. Saldature incomplete, ossia la lega brasante non ha bagnato completamente un reoforo.

3. Buchi di saldatura

4. Elementi estranei

Proprio quest'ultimi posso essere maggiormente pericolosi per i controlli effettuati con machine vision, in quanto imprevedibili.

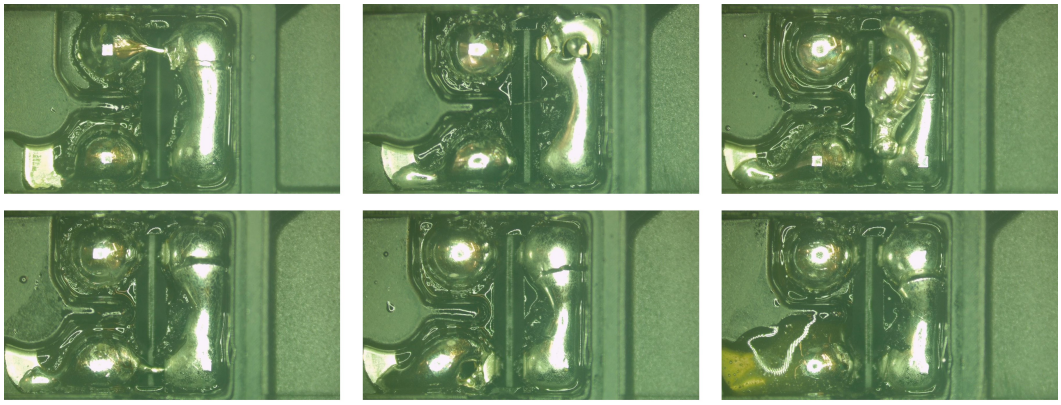


Figura 5.6: a) Cortocircuiti. b) Saldature incomplete e buchi di saldatura. c) Elementi estranei

5.3 Sistema attuale di controllo qualità della saldatura

Il sistema attualmente implementato per il controllo della conformità delle saldature si compone di due ispezioni visive:

1. Machine vision (ST70)
2. Controllo visivo operatore (ST80)

Questo permette al sistema di essere resiliente nel caso in cui uno dei due sistemi venisse meno. A livello logico come si può vedere in figura 5.7 il sistema segue la sequenza delle stazioni, ossia se il controllo di machine vision dopo l'analisi delle 4 saldature trova una non conformità, l'assemblato verrà scartato anche se l'operatore nella stazione ST80 non rileva la non conformità. Viceversa, se alla ST70 la telecamera assegna la conformità alle saldature, l'operatore alla ST80 può confermare l'esito precedente oppure assegnare la non conformità e quindi scartare l'assemblato.

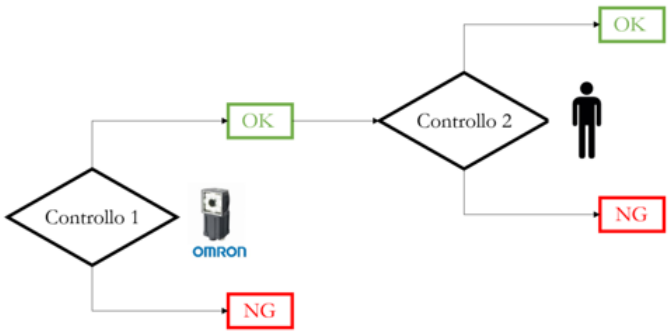


Figura 5.7: Schema logico del controllo qualità in linea delle saldature stagno

5.3.1 Ispezione con machine vision

Il sistema di machine vision è composto da un illuminatore flat dome IR e una telecamera FH Omron collegata ad un elaboratore che presenta un interfaccia grafica per lo sviluppo delle codice di analisi.
Il flat dome è un illuminatore a luce diffusa che si presenta come un pannello luminoso ed è l'evoluzione del dome, ossia della luce a cupola. I vataggi del flat dome rispetto al dome normale sono due:

1. Non si ha il vincolo di ingombro meccanico dato dalla struttura a cupola che aumenta la necessità di spazio nella stazione dedicata comportando anche il vincolo di minima distanza di lavoro della telecamera. (vedi figura 5.8)

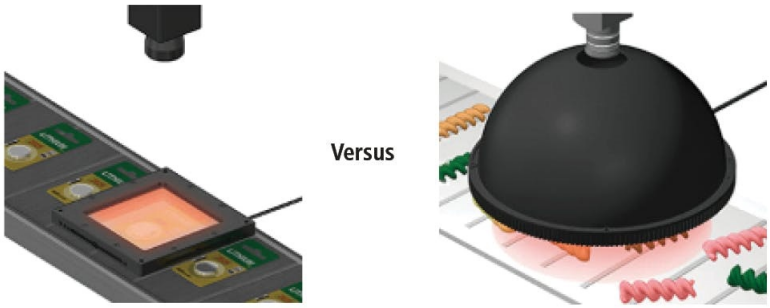


Figura 5.8: A sinistra è rappresentata la configurazione con flat dome a destra quella con il dome

2. Nell'immagine catturata dalla telecamera che usa un dome si avrà una zona di ombra data dal foro presenta nel dome per introdurre l'obiettivo. (vedi

figura 5.9)

Figura 5.9: A sinistra si ha la rappresentazione di una configurazione di illuminazione con dome. A destra invece l'esempio di zona d'ombra che si ottiene con il dome

Il flat dome per produrre la luce diffusa usa un vetro in cui al suo interno sono presenti delle particelle riflettenti che indirizzano la luce verso il basso, ossia verso il targhet. Questo metodo di illuminazione riesce a ridurre il disturbo che la riflessione produce nell'immagine, in particolare se si ispezionano parti metalliche o composte da pellicole trasparenti (figura 5.10).

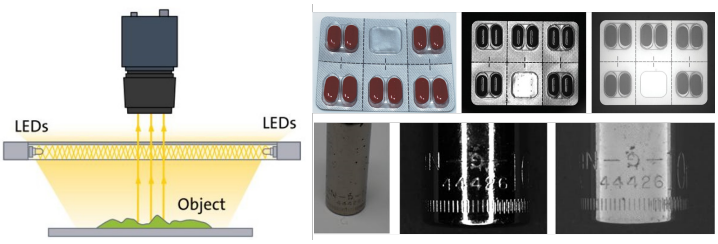


Figura 5.10: A sinistra si ha la rappresentazione di una configurazione di illuminazione con flatdome stilizzata. A destra invece si hanno degli esempi di immagini acquisite con illuminazione flat dome (a destra), mentre centralmente si hanno le stesse immagini ma acquisite con illuminazione diffusa ad anello

Nella ST70 abbiamo un flat dome che lavora nell'infrarosso, questo permette di ridurre al minimo la questione riflessi e il risultato è l'acquisizione di un immagine delle 4 saldature come in figura 5.11.

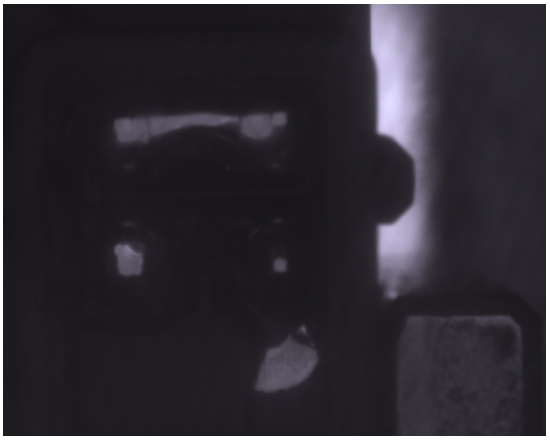


Figura 5.11: Immagine acquisita con l'illuminazione flat dome IR

La telecamera FH monocromatica (sensore ottico più l'obiettivo) e l'elaboratore fanno parte del pacchetto che Omron ha consigliato per lo sviluppo del controllo. Nell'elaboratore è stato sviluppato, mediante il software proprietario Omron, il programma che esegue il processamento delle immagini per estrarre le caratteristiche utili per classificare le saldature conformi o non. In seguito si elencano i passaggi che il tecnico ha deciso di inserire nel programma di controllo.

1. **Compensazione di forma:** Questa prima fase serve per orientare in modo corretto l'immagine acquisita a quella di riferimento con cui si è sviluppato il controllo. Per far ciò il tecnico sceglie il riferimento con cui allineare tutte le immagini ed in questo caso si è scelta la forma troncoconica in figura 5.11.

2. **Filtro binario:** Questo filtro come spiegato nel Capito 3 è un operazione Point-Point e trasforma l'immagine in binaria (bianco e nero)

3. **Determinazione delle aree di controllo:** In questa fase si sceglie le aree da analizzare a seconda dei difetti che si potrebbero verificare. In questo codice il tecnico ha scelto 6 aree di controllo e quindi si effettuano 6 controlli. Due per verificare che le saldature ST40 e ST50 siano isolate ossia che non ci sia stagno superfluo che potrebbe portare a cortocircuiti e 4 controlli per verificare che la lega brasante abbia bagnato correttamente tutta la piazzola e il reoforo di riferimento(vedi figura 5.12).

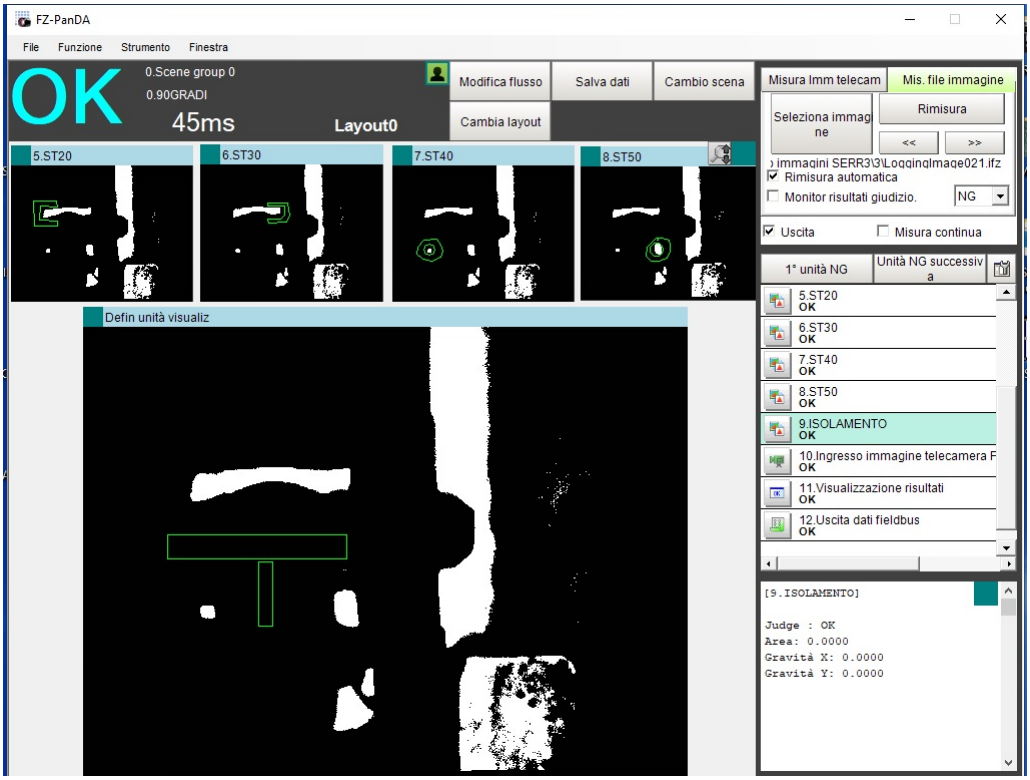


Figura 5.12: Interfaccia grafica del sistema di machine vision con in evidenza i 6 controlli

4. **Soglia di densità di bianco:** In quelle aree critiche determinate in precedenza si è scelta la soglia entro il quale la quantità di bianco è accettabile. Questa tecnica serve per effettuare la classificazione di ogni singolo controllo e se uno solo restituisce la non conformità, l'intera immagine sarà non conforme e quindi l'assemblato verrà scartato in ST90.

   Nelle due aree per il controllo dell'isolamento delle saldature ST50 e ST40 la quantità di bianco accettabile è praticamente nulla, questo perchè il bianco nell'immagine identifica la presenza di stagno e quindi un possibile corto circuito. Negli altri 4 controlli invece si ha la situazione inversa ossia lo stagno della saldatura si trova in ombra e quindi se è presente nell'immagine binaria corrisponda ad una zona nera. Se non fosse presente, la piazzola scoperta rifletterebbe la luce e quindi nell'immagine binaria sarebbe rappresentata da una zona di bianco. In queste aree di controllo la soglia di densità di pixels bianchi è più difficile da valutare infatti viene ripetuta-

mente aggiustata dal tecnico addetto,a causa delle molte variabili in gioco (un esempio, la riflettanza dello stagno).

5.3.2 Ispezione visiva operatore

Nella stazione ST80 avviene il controllo della conformità delle 4 saldature mediante il giudizio del operatore istruito della ST00. Per far si che l'operatore riesca ad effettuare l'ispezione nella stazione ST00 è stato inserito un monitor che restituisce in real time quello che il microscopio industriale cattura, quindi in questo si può contemporaneamente preassemblare il carter con i microinterruttori ed effettuare il controllo qualità. Se l'operatore rileva un anomalia della saldatura può scartare l'assemblato premendo un pulsante rosso sotto il monitor, se invece lo ritiene conforme non deve fare nulla.(vedi figura 5.13)

Figura 5.13: Setup controllo di qualità ST80

Per questo controllo la strumentazione utilizzata è essenziale, ossia un monitor collegato ad un microscopio industriale commerciale con le seguenti caratteristiche principali:

Sensore	16 MP CMOS
Dimensione Pixel	$1,335\mu m\ x\ 1,335\mu m$
interfacce	HDMI(Type A),USB(Type B),TF Card
Risoluzione HDMI	1920*1080p 60FPS
USB2.0(Drive free)	1920*1080@30 1280*720@30 640*480@30
Zoom Digitale	17X-108X

In dotazione al microscopio elettronico è presente un illuminatore a led ad anello ed è utilizzato per rendere l'immagine più insensibile a variazioni di luce ambientali. Con questa tipologia di illuminazione si ha il problema delle riflessioni ma l'operatore, se ben istruito, riesce ugualmente a riconoscere le non conformità, cosa che sarebbe impossibile per il sistema machine vision.
In conclusione, in questo capitolo si è analizzato lo stato attuale della linea, concentrandosi in particolare sul sistema di controllo qualità. Questo ha permesso di capire la soluzione migliore per implementare un sistema di raccolta dati che potesse al meglio soddisfare le esigenze per lo sviluppo del sistema di computer vision.

Capitolo 6

Sistema e metodologia di raccolta dati

Partendo dall'analisi precedente, fatta al Capitolo 5, si è deciso in accordo con i responsabili di reparto dell'azienda di sviluppare un sistema che potesse raccogliere i seguenti dati:

1. Le immagini delle 4 saldature di ogni singolo assemblato proveniente dal microscopio industriale presente nella stazione ST80.
   Questa scelta è stata fatta per diverse motivazioni, la principale è stata quella di volere replicare il controllo umano con il sistema di computer vision e confrontare i risultati di quest'ultimo con quelli ottenuti dal sistema di machine vision attualmente in linea. Il secondo motivo è quello di avere le immagini delle saldature che gli operatori determinano non conformi, così da effettuare formazione in caso di errore, oppure per capire gli esiti non concordanti tra operatore e sistema di machine vision. Ogni immagine è stata salvata con data e ora per avere un riferimento temporale che la potesse identificare in modo univoco.

2. Sviluppare un file CSV dove salvare per ogni immagine l'esito globale dato dal sistema di machine vision e l'esito globale dato dell'operatore sulla conformità delle saldature.

Tutto questo elencato doveva esser fatto senza interferire sul tempo ciclo della linea e senza alterare le condizioni di ispezione visiva dell'operatore.

6.1 Scelta del Hardware

La scelta del hardware oltre che tenere conto dei vincoli che si sono elencati precedentemente, si è basata anche sulla possibilità in seguito di implementare il sistema di computer vision per effettuarare la fase di testing del modello sviluppato e raccogliere dati delle sue prestazioni. Quindi per la scelta del hardware si sono poste le seguenti domande:

1. **Come ci si può interfacciare al microscopio industriale?**
   A questa domanda si è visto che il microscopio industriale può comunicare anche via USB (vedi caratteristiche nel capitolo precedente) con 3 diverse risoluzioni (1920*1080@30 1280*720@30 640*480@30). Questa cosa rende possibile catturare ogni frame prodotto dal microscopio mediante un codice scritto in Python con l'uso della libreria OpenCV, che è una libreria specifica per l'analisi delle immagini e video.

2. **Come raccogliere gli esiti sia del controllo effettuato dalla machine vision che dell'operatore dal PLC?**
   Il PLC della linea in questione può ricevere o trasmettere dati in due modalità: via EtherNet/IP o via bit (GPIO). Quindi l'hardware deve avere almeno l'interfaccia GPIO. Un esempio semplice ed economico sono i dispositivi Arduino.

3. **Cosa serve per implementare in linea un algoritmo di deep learning?**
   Il linguaggio maggiormente utilizzato per sviluppare reti neurali o algoritmi di machine learning è Python, in quanto viene utilizzato come base per i framework più famosi di deep learning come Tensorflow, Keras o Pytorch. Inoltre il sistema dovrà eseguire operazioni di analisi realtime molto velocemente.

Questo comporta di avere un sistema con GPU dedicata per effettuare analisi delle immagini velocemente e che possa compilare codici sviluppati in linguaggio Python.

Tenendo conto delle risposte alle precedenti domande si è deciso di utilizzare un dispositivo di edge computing chiamato Jetson Nano (versione 4GB B01)(vedi figura 6.1).

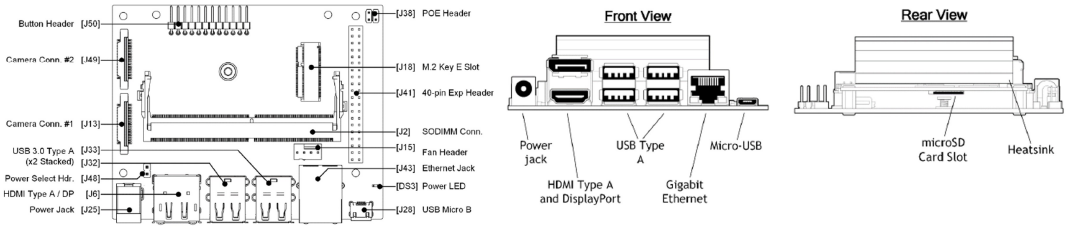


Figura 6.1: Jetson Nano B01

Questo dispositivo sviluppato da Nvidia è stato progettato proprio per semplici implementazioni di algoritmi di deep learning, ecco perchè presenta una GPU dedicata (128 Core Nvidia Maxwell) che ottimizza computazionalmente le analisi delle immagini, rendendo quest'ultime molto veloci ed adatte ad applicazioni real time. La Jetson Nano è basata sul sistema operativo Linux4Tegra Ubuntu 18.04 che è perfettamente compatibile con Python e con i framework Tensorflow, Keras o Pytorch, ciò da la possibilità di utilizzare algoritmi addestrati senza la necessità di riconvertirli in particolari formati.
A livello di interfaccia fisica I/O, il dispositivo presenta 40 pin GPIO, essenziali per comunicare con il PLC, 4 porte USB, utili per collegare altri dispositivi di supporto e il microscopio industriale, una porta HDMI, per dare la possibilità all'operatore di vedere nel monitor le 4 saldature per effettuare l'ispezione visiva precedentemente descritta, e infine la porta ethernet, utilizzata per interfacciare il pc aziendale così da prelevare i dati e aggiornare i codici senza interferire con il lavoro svolto dall'operatore in linea. (vedi figura 6.2)

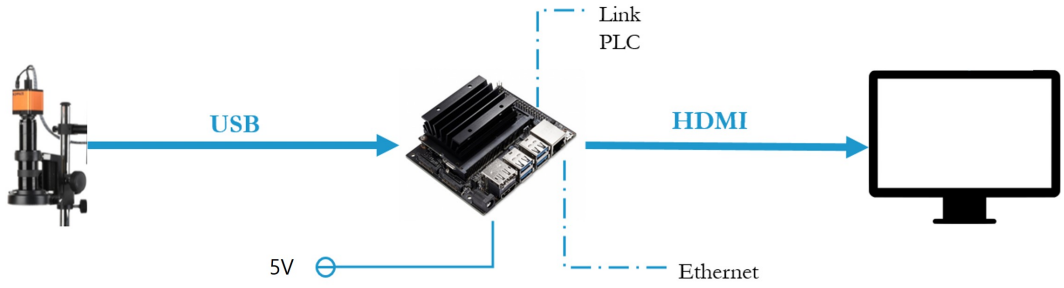


Figura 6.2: Sistema fisico per la Data Collection

Per acquisire gli esiti dal PLC c'è stato bisogno di introdurre un circuito ausiliario a causa delle differenti tensioni in gioco, in quanto il PLC lavora a 12 V mentre la Jetson Nano a 5V. Il circuito in figura 6.3 è il circuito di interfaccia PLC-Jetson Nano che si è utilizzato per ogni canale di acquisizione. L'elemento chiave del circuito è il relè che in condizioni normalmente chiuso (1-2), non permette il passaggio di corrente e quindi il pin GPIO legge lo stato 0, mentre quando il PLC fa passare corrente nel relè (invia il segnale), il relè passa allo stato normalmente aperto (3-2), permettendo il passaggio di corrente e quindi il pin legge lo stato 1.

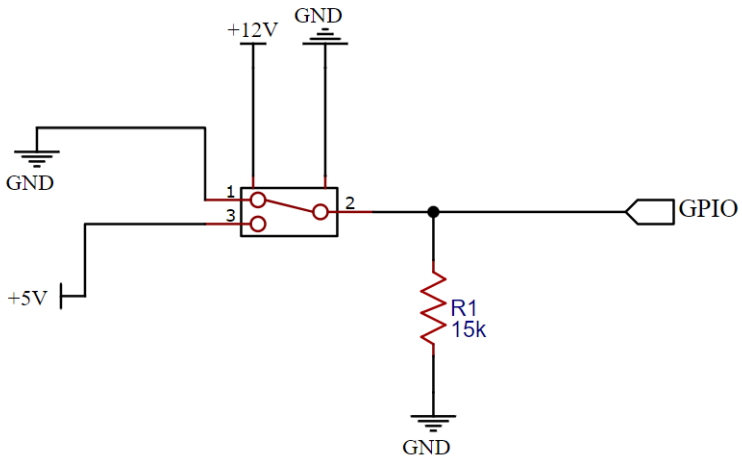


Figura 6.3: Circuito di interfaccia tra la Jetson Nano e il PLC

Dopo diverse analisi si è deciso di acquisire 4 segnali dal PLC che sono il trigger di rotazione tavola, utile per capire quando si hanno delle nuove saldature da analizzare, l'esito dell'analsi fatta dalla stazione di machine vision (ST70) e l'esito del controllo dell'operatore (ST80). Per raccogliere l'esito del controllo del

sistema di machine vision si sono utilizzati due canali distinti, uno per quando le saldature sono conformi e uno per quando non lo sono, questo ha permesso di capire anche se il sistema di machine vision è momentaneamente disattivato o non funzionante. Fisicamente come si vede in figura 6.4, si è sono costruite 5 circuiti con 5 relè per ogni circuito rispettando lo schema di figura 6.3. Si è aggiunto un canale per possibili acquisizioni future.

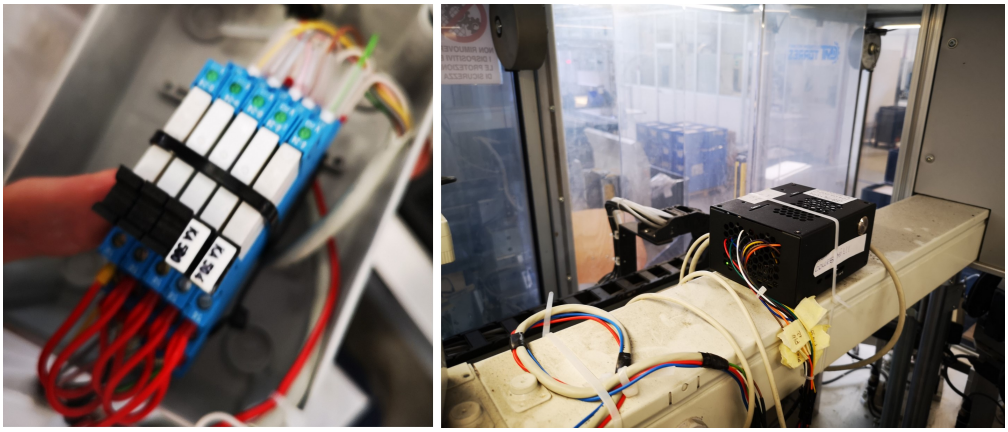


Figura 6.4: A sinistra si hanno i 5 relè utilizzati per l'interfaccia PLC-Jetson Nano. A destra la Jetson Nano nel suo case nero installata sulla linea automatica

Finito il processo di scelta e di assemblaggio del'hardware si è passati allo sviluppo del codice che possa permettere tutto quello detto precedentemente.

6.2 Codice di Data Collection

Il codice Python sviluppato per la raccolta dati si compone principalmente di due grandi funzioni: la funzione che permette il salvataggio delle immagini delle 4 saldature di ogni assemblato e la funzione che raccoglie gli esiti dei controlli qualità (machine vision e operatore) e li salva in un file CSV (schema logico in figura 6.5).

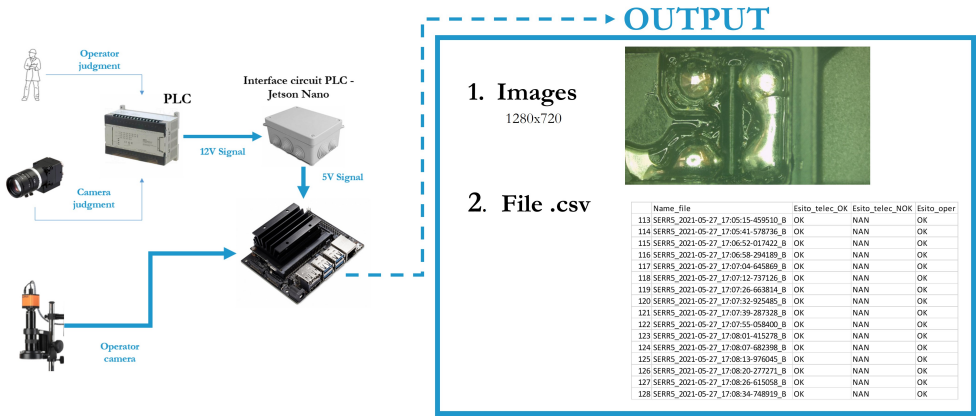


Figura 6.5: Schema logico del codice data collection

Prima di sviluppare la funzione di salvataggio delle immagini è stato necessario capire in che risoluzione salvarla e che frame acquisire in tutta la permanenza nell'assemblato sulla stazione ST80.

Per il primo punto si aveva la possibilità di scegliere 3 tipologie di risoluzione, senza andare ad alterare i frame al secondo (vedi caratteristiche microscopio industriale sezione 5.2.2) e si è scelta la risoluzione 1280x720 a 30 fps, in quanto era la risoluzione del monitor. Questo ha permesso di non alterare l'immagine vista dall'operatore e quindi di non alterare gli esiti delle ispezioni. Riguardo il secondo punto, ossia quale frame salvare si è dovuto fare un ragionamento tendo conto del tempo ciclo della linea e delle operazioni eseguite in quella stazione. Per far ciò si è registrato un video che riprendesse tutto ciò che il microscopio industriale riusciva a vedere per un determinato periodo di produzione. In seguito si è analizzato un campione del video trovando il valore del MSE tra il frame corrente e il precedente come si vede nel grafico in figura 6.6.

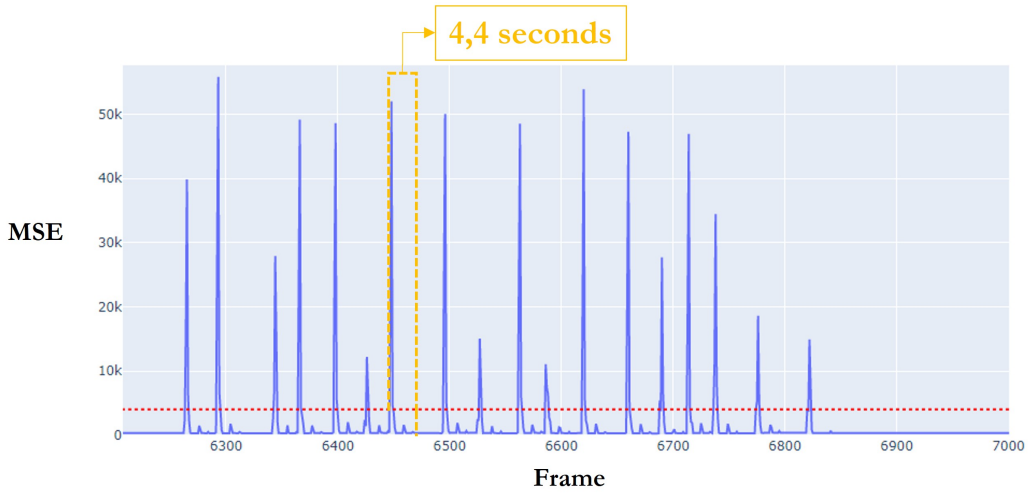


Figura 6.6: Grafico MSE-numero frame risultante dall'analisi del video di produzione ST80

Si può notare che ci sono due tipologie di picchi di MSE divisi da una linea tratteggiata rossa. I picchi di MSE maggiori di 5000 stanno ad indicare una enorme differenza tra il frame attuale e il precedente ed infatti a quel picco corrisponde il momento in cui la tavola sta ruotando. I picchi sotto la linea rossa invece indicano che tra una rotazione di tavola e l'altra abbiamo periodicamente un piccolo spostamento dell'assemblato, che analizzando le operazioni della stazione, corrisponde al momento in cui il connettore per il collaudo elettrico vieni inserito nel carter. L'acquisizione dell'immagine delle saldature di ogni assemblato si è decisa di farla dopo il secondo picco in quanto l'immagine è stabile e tutte e 4 le saldature sono correttamente visibile. Calcolando la relazione che intercorre tra i frame e i secondi (FPS) si è riuscito a capire dopo quanto tempo acquisire l'immagine una volta ottenuto il trigger della rotazione tavola.
Questi ragionamenti in seguito sono stati tradotti in codice Python utilizzando la libreria OpenCV la quale contiene molte funzioni per gestire file di immagine, tra cui quella per il salvataggio delle immagini.
L'altra funzione invece tratta la raccolta dei segnali provenienti dal PLC. Per lo sviluppo di questo codice si sono utilizzate altre librerie come Pandas, per la creazione e l'accrescimento del file CSV, e la libreria GPIO per gestire i pin che sono collegati con i relè. Per l'acquisizione dei segnali si è strutturata una

logica che filtra il segnale reale dal possibile rumore fatto dai rimbalzi dei relè che potrebbero compromettere l'attendibilità degli esiti. La logica utilizzato per l'acquisizione dei segnali è rappresentata nel diagramma di flusso in figura 6.7 che usa diverse variabili di supporto per funzionare.

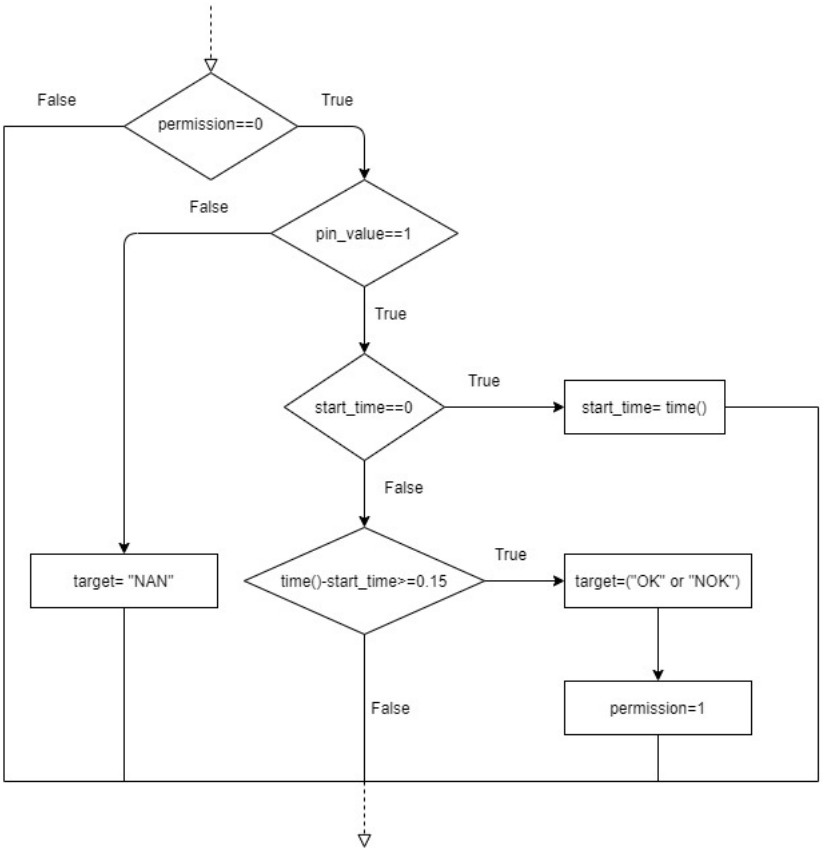


Figura 6.7: Diagramma di flusso per logica anti-rimbalzo relè

In sintesi, il segnale viene acquisito e ritenuto attendibile se e solo se è presente per almeno 0.15 secondi, che è un tempo sufficientemente alto per essere sicuri che non venga dato da rimbalzi del relè. Quando il segnale viene acquisito oltre che a salvare l'esito in una variabile, si blocca la possibilità di lettura dello stato del pin GPIO per non sovrascrivere le informazioni raccolte. Ad ogni giro tavola, infine, viene aggiunta una riga al file CSV dove si raccolgono tutti gli esiti con la seguente struttura (vedi figura 6.8): nella prima colonna è presente il nome dell'immagine con data e ora, nella seconda il segnale della conformità data dalla stazione ST70 (può essere "NAN", non ha ricevuto il segnale, o "OK", il segnale filtrato è stato

acquisito), nella terza il segnale della non conformità data sempre dalla stazione ST70 (”NAN o ”NOK”) e la quarta il segnale dell'esito dell'operatore (”OK” o ”NOK”). Se per una stessa immagine è presente la stringa ”NAN” sia nella seconda che nella terza colonna, allora il sistema di machine vision potrebbe aver avuto problemi o esser stato disattivato. Avvenuto il salvataggio le variabili vengono resettate, pronte per acquisire nuovi segnali provenienti dal successivo controllo di altre saldature.

Un'osservazione importante è che l'esito del controllo di machine vision che nel file CSV è delegato ad una precisa immagine, nella realtà corrisponde all'esito dell'immagine successiva. In fase di pulizia dei dati, questa imprecisione dovuta alla natura del processo di assemblaggio, verrà corretta eseguendo un semplice shift di 1 alle colonne 2 e 3. Il file CSV, come detto in precedenza, ogni giro tavola si arricchisce di una riga, ma nella realtà si è deciso di aggiornarlo ogni 60 secondi per non appesantire troppo il carico computazionale del dispositivo.

	Name_file	Esito_telec_OK	Esito_telec_NOK	Esito_oper
113	SERR5_2021-05-27_17:05:15-459510_B	OK	NAN	OK
114	SERR5_2021-05-27_17:05:41-578736_B	OK	NAN	OK
115	SERR5_2021-05-27_17:06:52-017422_B	OK	NAN	OK
116	SERR5_2021-05-27_17:06:58-294189_B	OK	NAN	OK
117	SERR5_2021-05-27_17:07:04-645869_B	OK	NAN	OK
118	SERR5_2021-05-27_17:07:12-737126_B	OK	NAN	OK
119	SERR5_2021-05-27_17:07:26-663814_B	OK	NAN	OK
120	SERR5_2021-05-27_17:07:32-925485_B	OK	NAN	OK

Figura 6.8: Tabella risultante dei esiti raccolti

Capitolo 7

Modello di Computer Vision

Una volta strutturata la parte per la raccolta dati si può parlare del cuore del sistema di computer vision, ossia dell'algoritmo sviluppato per eseguire la misurazione della qualità delle giunzioni saldate e le tecniche a supporto per eseguire la classificazione tra conforme e non. Per lo sviluppo del modello di computer vision si sono seguiti gli steps detti al Capitolo 4 ossia:

1. Sviluppare un sistema che permettesse di misurare oggettivamente la conformità della saldatura di ogni singola giunzione, che fosse facilmente interpretabile e che fosse robusto alle variabili di disturbo ambientali (modeste variazioni di luce e di posizione).

2. Nella seconda fase si è effettuata la classificazione binaria dei giunti utilizzando delle specifiche soglie per ogni saldatura. Questo ha permesso di ricavare un esito globale per determinare la conformità o meno dell'assemblato.

3. Il valore di ogni soglia è stato scelto prendendo un campione di riferimento di immagini di saldature valutate da una giuria di esperti. In seguito mediante l'ottimizzazione combinatoria si sono trovate le combinazioni che massimizzano il macro F1-score.

In seguito si tratteranno in dettaglio tutti questi punti. Per tutto il lavoro svolto si è utilizzato il linguaggio Python ed in particolare per lo sviluppo delle reti

neurali si è utilizzata la libreria Keras, che usa in background Tensorflow, per l'analisi dei dati tabulari si sono utilizzati i pacchetti Pandas e Numpy ed infine per la visualizzazione dei dati si sono utilizzate le librerie Matplotlib, Seaborn ed Plotly.

7.1 Algoritmo di misura

Per misurare la qualità della saldatura di ogni giunto si è sviluppato un algoritmo di anomaly detection per ogni stazione utilizzando una rete neurale convoluzione che rientra nelle famiglie dei Convolutional Autoencoder (CAE). Il principio che sta alla base di questo algoritmo è quello di allenare la rete neurale ad estrarre le informazioni più importante dall'immagine (fase di encoder) ed inseguito ricostruirla in maniera più simile possibile all'originale.
Dopo molte ricerche e molti test, si è sviluppato l'algoritmo di Convolutional Autoencoder in figura 7.1 che riesce ad ricostruire l'immagine in modo accettabile per l'analisi.

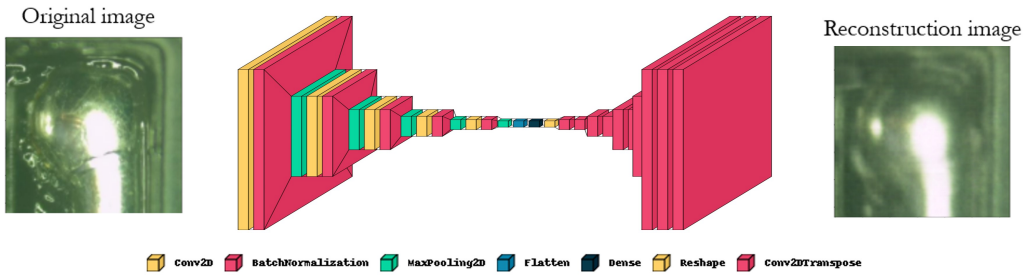


Figura 7.1: Struttura CAE usata in questa trattazione

7.1.1 Data Preparation

Prima di entrare nel merito dell'algoritmo si spiegheranno i passaggi preliminari di elaborazione che si sono eseguiti per avere le immagine in un formato idoneo alla rete neurale.
L'immagine acquisita, come precedentemente detto, è nella codifica RGB con dimensioni (720,1280,3). Mediante delle operazioni di taglio si sono ricavate le 4 immagini delle giunzioni saldate nel formato (360,360,3) e in seguito si sono

scalate le dimensioni a (128,128,3) per ottenere un'immagine con minor pixels in quanto maggiore è il numero di pixels, maggiore è il costo computazione della rete neurale. Ultima fase è la normalizzazione dei pixels rispetto al valore massimo che essi possono avere che in questo caso specifico, visto che l'immagine è a 8 bit, il valore sarà 255 (vedi sezione 3.1). La divisione per 255 porta il valore di ogni pixel tra 0 e 1 rendendo l'apprendimento della rete molto più veloce.

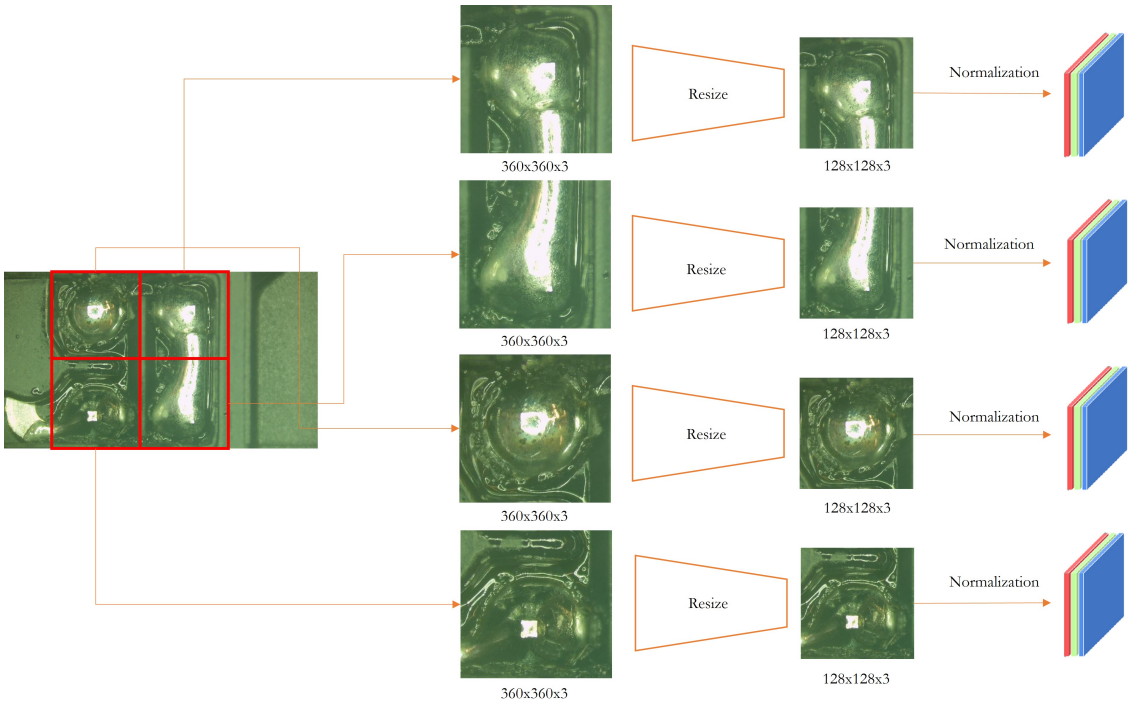


Figura 7.2: Fase di preprocessamento delle immagini

La fase di data preparation è utilizzata sia in fase di creazione dei dataset per addestrare la rete di anomaly detection, ma anche nella fase di analisi in real time. Quindi gli steps di preprocessamento dell'immagine non devono essere troppi in quanto questo porta un aumento del carico computazione che il sistema di edge computing dovrà sostenere in real time, ma abbastanza da poter avere una rete neurale che possa eseguire le analisi in modo efficace e veloce.

7.1.2 Encoder

La prima parte dell'algoritmo è l'encoder ed è composto da 17 layers divisi in 5 strati di convoluzione, 5 di Batch Normalization, 5 di Max Pooling, uno di

Flatten e l'ultimo di percettroni (chiamato Dense). Per i layers di convoluzione si è utilizzata la funzione di attivazione ELU che ha portato notevoli riduzioni dei tempi per la fase di addestramento. Per maggior chiarezza in figura 7.3 sono riportate le specifiche di ogni layer del encoder:

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 128, 128, 128)	3584
batch_normalization_1 (Batch	(None, 128, 128, 128)	512
max_pooling2d_1 (MaxPooling2	(None, 64, 64, 128)	0
conv2d_2 (Conv2D)	(None, 64, 64, 64)	73792
batch_normalization_2 (Batch	(None, 64, 64, 64)	256
max_pooling2d_2 (MaxPooling2	(None, 32, 32, 64)	0
conv2d_3 (Conv2D)	(None, 32, 32, 32)	18464
batch_normalization_3 (Batch	(None, 32, 32, 32)	128
max_pooling2d_3 (MaxPooling2	(None, 16, 16, 32)	0
conv2d_4 (Conv2D)	(None, 16, 16, 16)	4624
batch_normalization_4 (Batch	(None, 16, 16, 16)	64
max_pooling2d_4 (MaxPooling2	(None, 8, 8, 16)	0
conv2d_5 (Conv2D)	(None, 8, 8, 8)	1160
batch_normalization_5 (Batch	(None, 8, 8, 8)	32
max_pooling2d_5 (MaxPooling2	(None, 4, 4, 8)	0
flatten_1 (Flatten)	(None, 128)	0
dense_1 (Dense)	(None, 128)	16512

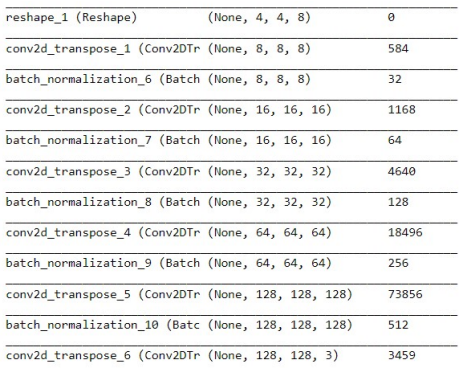
Figura 7.3: Struttura Encoder

Prendendo come riferimento la colonna "Output Shape", si può notare la progressiva riduzione della dimensionalità dell'immagine fino ad arrivare allo strato denso che presenta esclusivamente 128 features. Un'altra cosa importante è che non tutti gli strati hanno dei paramentri da addestrare, per esempio, il Max Pooling non presenta nessun paramento che va allenato in quanto è un semplice Local Operation non lineare di selezione del massimo. Tra lo strato di convoluzione e quello di max pooling si ha un ulteriore layer chiamato Batch Normalization che serve a normalizzare i valore dei pesi di ogni filtro, questo permette un aumento della velocità di allenamento in quanto fa lavorare il gradiente in condizioni ottimali.

7.1.3 Decoder

La seconda parte del Convolutional Autoencoder si chiama decoder ed è composto da 12 layers. Rispetto al encoder non presentano gli strati di Max Pooling in

quanto in questo caso le dimensioni non dovranno diminuire ma dovranno cresce fino a che non si ricomporrà la figura con le dimensioni originali (vedi figura 7.4).



reshape_1 (Reshape)	(None, 4, 4, 8)	0
conv2d_transpose_1 (Conv2DTr	(None, 8, 8, 8)	584
batch_normalization_6 (Batch	(None, 8, 8, 8)	32
conv2d_transpose_2 (Conv2DTr	(None, 16, 16, 16)	1168
batch_normalization_7 (Batch	(None, 16, 16, 16)	64
conv2d_transpose_3 (Conv2DTr	(None, 32, 32, 32)	4640
batch_normalization_8 (Batch	(None, 32, 32, 32)	128
conv2d_transpose_4 (Conv2DTr	(None, 64, 64, 64)	18496
batch_normalization_9 (Batch	(None, 64, 64, 64)	256
conv2d_transpose_5 (Conv2DTr	(None, 128, 128, 128)	73856
batch_normalization_10 (Batc	(None, 128, 128, 128)	512
conv2d_transpose_6 (Conv2DTr	(None, 128, 128, 3)	3459

Figura 7.4: Struttura Decoder

In questa fase, si sono utilizzati particolari layers che usano la convoluzione trasposta, ossia un operazione che riesce ad espandere la dimensione dell'immagine attuando contemporaneamente il principio della convoluzione. Tra questi layer di convoluzione trasposta c'è sempre un layer di bach normalizzation.
L'output del decoder sarà un immagine con le stesse dimensione dell'immagine di input quindi (128,128,3).

7.1.4 Fase di Training

Una volta strutturato l'algoritmo si passa al suo addestramento. Visto che si è utilizzato il Convolutional Autoencoder come strumento di anomaly detection, per l'addestramento di quest'ultimo si utilizzeranno esclusivamente le immagini di giunti saldati conformi.
Per ogni singola saldatura si è allenato un CAE con le corrispettive immagini conformi, quindi complessivamente si sono sviluppate 4 reti neurali convoluzionali avendo a disposizione 4 dataset delle singole saldature conformi. I dataset dell'ultima versione (V5) di questi algoritmi sono composti rispettivamente da 4855 immagini per il giunto ST20, da 4203 immagini per il giunto ST30, da 4028 immagini per il giunto ST40 e da 4145 immagini per il giunto ST50. Ogni dataset è stato diviso in altri due dataset uno chiamato training set, corrisponde

al 80% del totale, e mentre l'altro prende il nome di validation set, corrisponde al 20% del totale. Il training set è l'insieme di tutte le immagini che vengono utilizzate per effettivamente eseguire l'apprendimento, ossia l'aggiornamento dei pesi dei filtri. Il validation set sono l'insieme delle immagini utilizzate per valutare l'apprendimento ad ogni epoca e quindi non prendono effettivamente parte all'addestramento.
Si è scelta come funzione di perdita MSE e come sistema di ottimizzazione Adam con $\alpha = 10^{-3}$, $\beta_1 = 0.9$, $\beta = 0.999$, $e = 10^{-8}$, inoltre per limitazioni di capacità di GPU (si è usata una GeForce GTX 1060), si è addestrato con il batch size di 32, ossia tutto il training set è stato diviso in lotti da 32 immagini che successivamente venivano passati alla rete per l'addestramento. Questo processo è stato eseguito per 100 epoche, un epoca corrisponde ad un addestramento su la totalità del training set.

```
Epoch 75/100
122/122 [==============================] - 27s 224ms/step - loss: 0.0035 - accuracy: 0.9663 - val_loss: 0.0042 - val_accuracy:
0.9652
Epoch 76/100
122/122 [==============================] - 27s 224ms/step - loss: 0.0035 - accuracy: 0.9670 - val_loss: 0.0045 - val_accuracy:
0.9665
Epoch 77/100
122/122 [==============================] - 27s 224ms/step - loss: 0.0035 - accuracy: 0.9664 - val_loss: 0.0042 - val_accuracy:
0.9691
Epoch 78/100
122/122 [==============================] - 27s 223ms/step - loss: 0.0035 - accuracy: 0.9665 - val_loss: 0.0043 - val_accuracy:
0.9681
Epoch 79/100
122/122 [==============================] - 27s 222ms/step - loss: 0.0034 - accuracy: 0.9665 - val_loss: 0.0045 - val_accuracy:
0.9664
Epoch 80/100
122/122 [==============================] - 27s 223ms/step - loss: 0.0034 - accuracy: 0.9666 - val_loss: 0.0043 - val_accuracy:
0.9676
Epoch 81/100
122/122 [==============================] - 27s 223ms/step - loss: 0.0035 - accuracy: 0.9669 - val_loss: 0.0043 - val_accuracy:
0.9677
Epoch 82/100
122/122 [==============================] - 27s 223ms/step - loss: 0.0034 - accuracy: 0.9674 - val_loss: 0.0043 - val_accuracy:
0.9680
Epoch 83/100
122/122 [==============================] - 27s 223ms/step - loss: 0.0034 - accuracy: 0.9670 - val_loss: 0.0042 - val_accuracy:
0.9683
Epoch 84/100
122/122 [==============================] - 27s 223ms/step - loss: 0.0034 - accuracy: 0.9670 - val_loss: 0.0042 - val_accuracy:
0.9676
```

Figura 7.5: Esempio di addestramento prendendo come riferimento il CAE per la saldatura ST20

In seguito si possono vedere le curve di apprendimento dei 4 algoritmi.

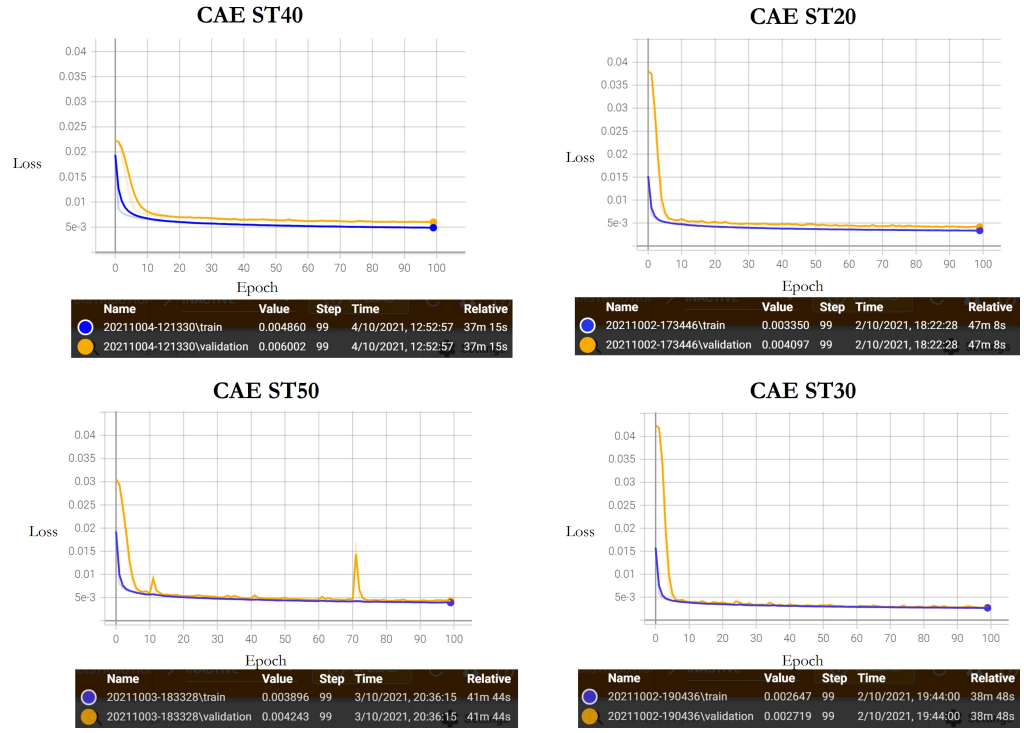


Figura 7.6: Quattro grafici loss-epoche che descrivono le curve di apprendimento di ogni CAE addestrato

Si può notare che in ogni grafico della figura 7.6 si hanno due curve che monitorano l'apprendimento, una blu che rappresenta il valore medio di loss che il modello calcola su tutto il training set ad ogni epoca, mentre quella arancione rappresenta la stessa cosa della precedente ma sul validation set.
Si può affermare che l'apprendimento è avvenuto correttamente se la curva del training set ha un valore di loss molto basso (valore di MSE basso) e se la curva del validation set, a fine addestramento è quasi coincidente con quella del training set. Il primo motivo è legato al fatto che minore è il valore di MSE migliore è la ricostruzione dell'immagine, mentre il secondo significa che la rete è riuscita a generalizzare il modello e quindi riesce a ricostruire le immagini che non sono state utilizzate in fase di addestramento. In termini tecnici, si dice che la rete non presenta overfitting. Si può vedere dalla figura 7.6 che tutte e 4 le reti hanno imparato a ricostruire le immagini in quanto il valore di MSE è molto basso, mentre si ha un leggero overfitting nei CAE ST20 e ST40.
Si può notare dalle curve di apprendimento che la loss tende a stabilizzarsi già

dopo 10 epoche, mentre in seguito tende lentamente a decrescere. Per le risorse a disposizione, si è scelto di addestrare le reti con 100 epoche che in media corrisponde utilizzando la GeForce GTX 1060 a 40 minuti per ogni rete.

7.1.5 Metodo di misura

Una volta allenati correttamente gli algoritmi di anomaly detection, questi sono capaci di ricostruire bene le immagini dei giunti conformi restituendo un valore di MSE molto simile al valore finale che presenta la curva di validation. Se invece proviamo ad analizzare un'immagine di un giunto con delle imperfezioni, l'algoritmo in prossimità dei difetti avrà problemi nella ricostruzione e questo porta l'aumento del valore di MSE globale.
Per capire meglio questo concetto e come lavora il sistema di anomaly detection si è sviluppata una maschera dove dove ogni suo pixel ha come valore l'errore quadratico medio di ricostruzione (vedi figura 7.7).

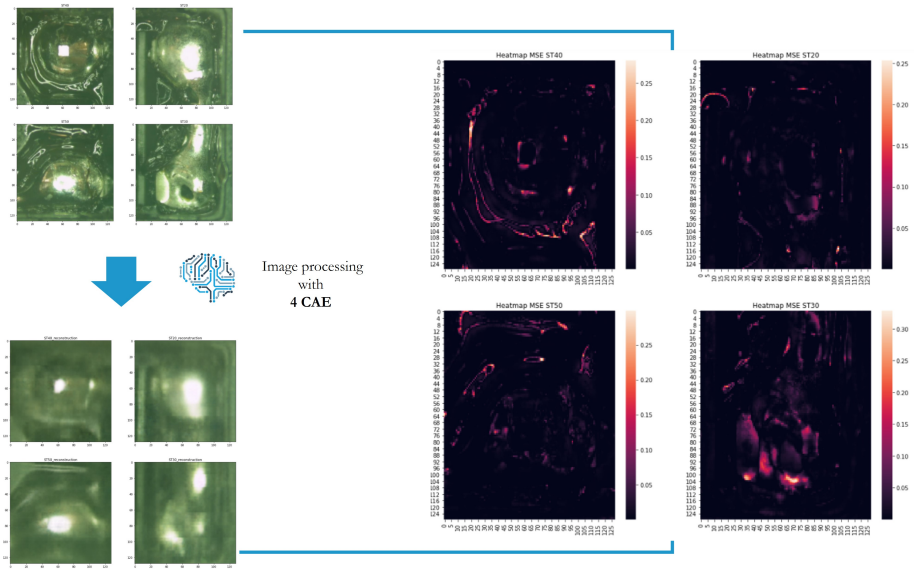


Figura 7.7: Processo logico per sviluppare la maschera di MSE, utile per facilitare la comprensione del lavoro svolto dagli algoritmi di anomaly detection

In seguito per ogni stazione verranno presentate 3 immagini di giunzioni conformi con la ricostruzione effettuata dalla specifica rete neurale e con la corrispettiva maschera di MSE.

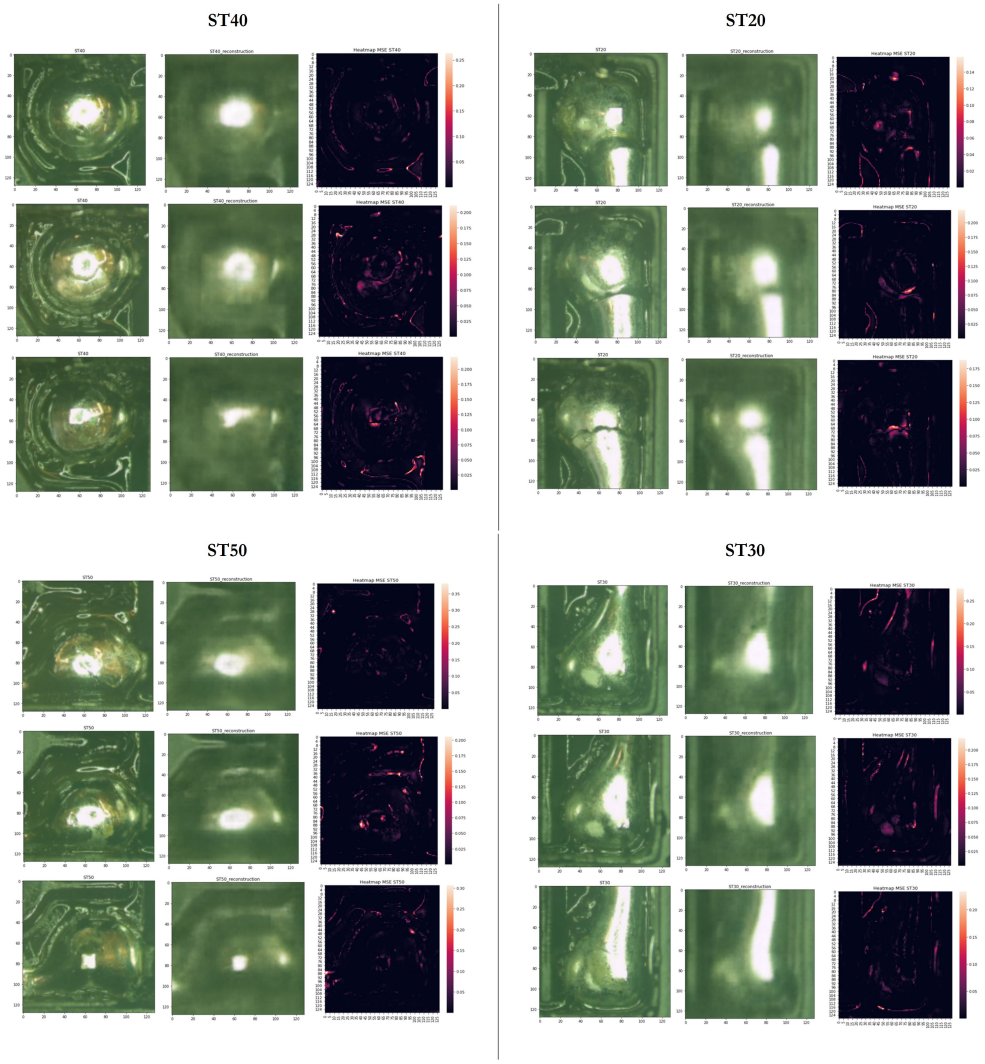


Figura 7.8: Esempi di giunzioni conformi per ogni stazione di saldatura

Si può notare dalla figura 7.8 come per le saldature conformi, l'immagine ricostruita è ben fatta a meno del rumore di fondo che ovviamente il sistema di reti neurali non è riuscita a modellizzare. Questo è ben visibile dalla maschera, dove le zone più chiare ricalcano in buona parte il rumore di fondo.
Gli MSE globali di queste precise immagini sono i seguenti:

Valori di MSE- Giunti Conformi							
ST20		ST30		ST40		ST50	
1	**0,0034**	1	**0,0049**	1	**0,0042**	1	**0,0044**
2	**0,0036**	2	**0,0062**	2	**0,0050**	2	**0,0052**
3	**0,0032**	3	**0,0035**	3	**0,0051**	3	**0,0050**

Ora si vedrà la differenza di ricostruzione con delle immagini non conformi. Si presenteranno in seguito 3 immagini non conformi per ogni stazione con le relative ricostruzioni e maschere.

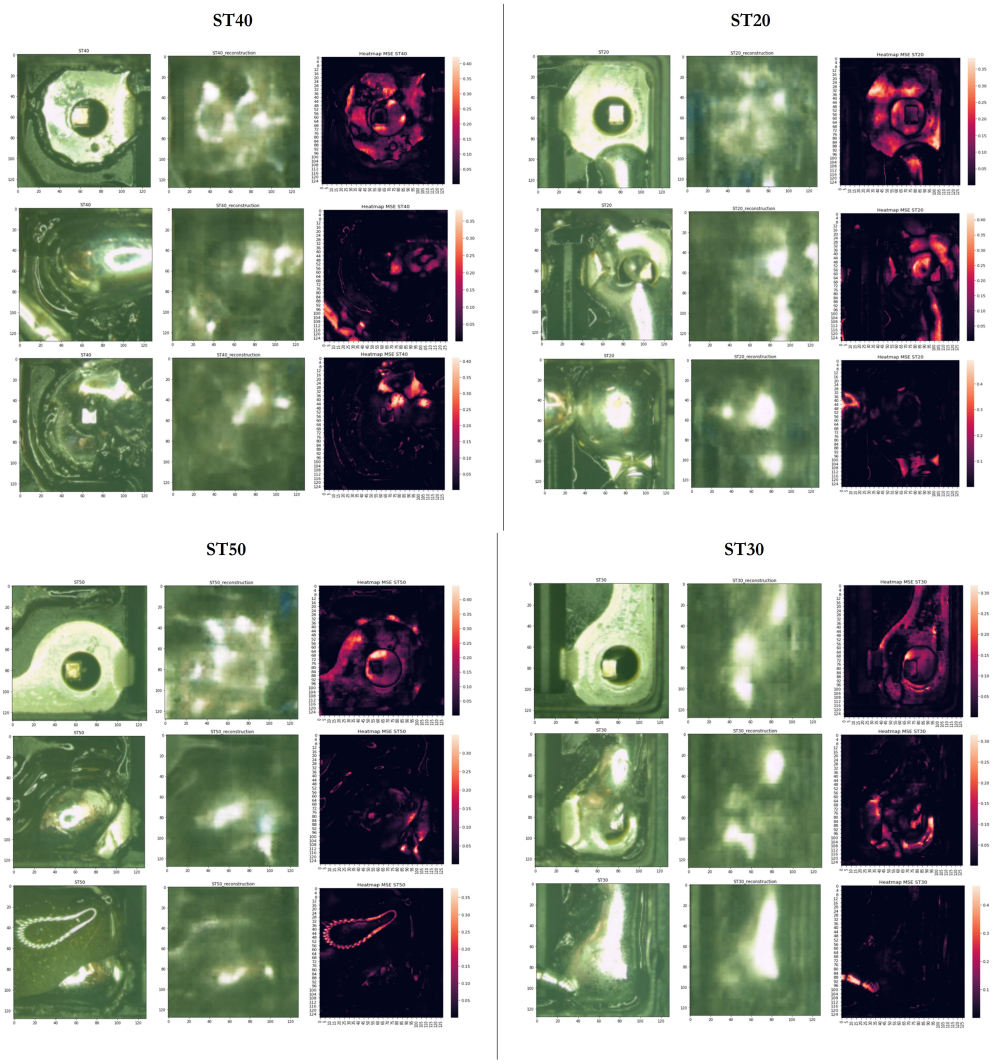


Figura 7.9: Esempi di giunzioni non conformi per ogni stazione di saldatura

Come si può notare in figura 7.9, il caso estremo di non conformità è l'assenza della saldatura che i sistemi di anomaly detection ricostruiscono in maniera pressoché randomica ed infatti nella maschera saranno presenti molte zone chiare che comportano un aumento consistente del valore MSE globale. Per difetti meno evidenti, come cortocircuiti, saldature incomplete e corpi estranei, il sistema riesce sempre ad identificarli (lo si può notare nella maschera corrispondente) anche se i valori di MSE globali non saranno dello stesso ordine di grandezza delle non

saldate.

In seguito sono visibili i rispettivi MSE per ogni difetto visto nelle immagine 7.9:

Valori di MSE- Giunti NON Conformi							
ST20		ST30		ST40		ST50	
1	**0,0458**	1	**0,0277**	1	**0,0391**	1	**0,0375**
2	**0,0324**	2	**0,0145**	2	**0,0187**	2	**0,0099**
3	**0,0158**	3	**0,0082**	3	**0,0211**	3	**0,0102**

7.2 Classificazione

Per effettuare la classificazione si sono utilizzate le immagini raccolte in una giornata tipica di produzione. A tutte le immagini è stato assegnato manualmente da una giuria l'esito globale, con la logica che in presenza di una singola non conformità in uno dei 4 giunti saldati, tutta l'immagine presenta la non conformità. Per creare gli esiti della giuria ci si è aiutati con gli esiti raccolti dal PLC e in caso di giunti saldati dubbi si è ricorso alla valutazioni di esperti del reparto qualità o mediante la consultazione della normative IPC J-STD-001G IT (Requisiti per la Brasatura degli Assemblaggi Elettrici ed Elettronici) e del manuale del IIS (ISTITUTO ITALIANO DELLA SALDATURA).

La scelta di dare un esito globale al posto che per ogni signola saldatura è per rispettare lo standard utilizzato dagli altri controlli in linea e anche per minimizzare il tempo speso nella classificazione manuale.

Come riferimento si è preso in esame la produzione a partire dalle 9:19:47 del 22/06/2021 fino alle 14:43:39 del 23/06/2021 in cui si sono assemblati 10191 carter con i due rispettivi microinterruttori. Nei grafici in figura 7.10 sono riportati dei punti che rappresentano i valori di MSE che ogni singolo CAE ha elaborato per la sua rispettiva saldatura che è identificata univocamente con l'ID, che rispetta la sequenzialità temporale mentre il colore rappresenta l'esito globale che la giuria ha determinato per l'assemblato.

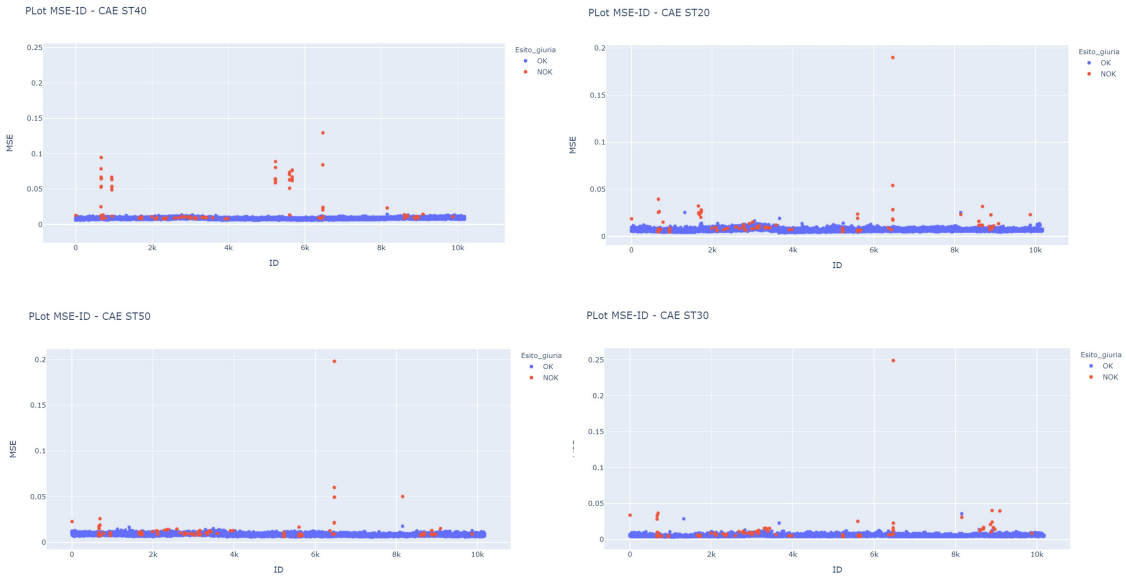


Figura 7.10: Grafici MSE-ID del lotto 22/06/2021-23/06/2021. Analisi di ogni saldatura è avvenuta con il suo corrispettivo algoritmo di anomaly detection.

In questa fase si è preferito identificare ogni singola immagine con un ID al posto che con data e ora in quanto il grafico si sarebbe arricchito di intervalli vuoti che in questa trattazione non porta nessuna informazione aggiuntiva.
Per come è stato pensato e sviluppato l'algoritmo di misura, si è trasformato un problema di controllo qualità delle immagini in un problema univarito che dipende esclusivamente dal valore assoluto del MSE. Proprio per il motivo precedente, la classificazione è possibile effettuarla con un soglia lineare, ossia tutte le saldature con valore di MSE maggiore della soglia di riferimento saranno identificate come non conformi (NOK), al contrario se il loro valore si trova sotto a quello di soglia allora saranno conformi (OK). Questo semplice metodo di classificazione permette una comprensione immediata anche alle persone meno esperte in quanto simile alla logica delle carte di controllo processo.
In questa fase del lavoro ci si è concentrati nel cercare la migliore combinazione delle 4 soglie (una per ogni stazione di saldatura e quindi una per ogni algoritmo di misura) che portasse risultati più vicini possibili a quelli della giuria. Per scegliere correttamente le 4 soglie si è sviluppata una tecnica di ottimizzazione combinatoria prendendo spunto dal DOE (Design of Experiment). Dall'analisi

statistica dei 4 grafici visti in figura 7.10 si sono potute separare le distribuzioni dei giunti OK da quelli NOK per ogni singola stazione. Il risultato è visibile in figura 7.11 dove si sono utilizzati i boxplot che è una visualizzazione sintetica ma esplicativa delle grandezze statistiche del campione.

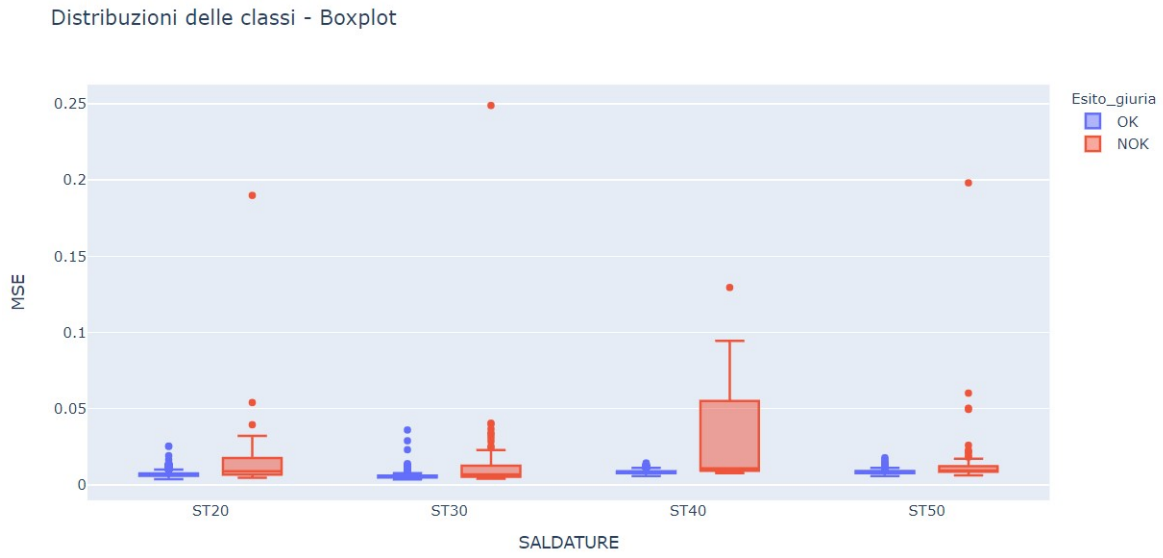


Figura 7.11: Rappresentazione mediante boxplot delle grandezze statistiche del capione di riferimento

Si è deciso di prendere per ogni saldatura l'intervallo tra la mediana e il 3 quartile della distribuzione della classe dei difetti (NOK) e dividerlo in 10 valori ciascuno. Utilizzando questi 40 valori, 10 per ognuna delle 4 stazioni di saldatura, si sono create tutte le possibili disposizioni (combinazioni ordinate), ottenendo così 10000 combinazioni di soglie. I relativi intervalli sono i seguenti:

Stazione	**Intervallo MSE**
ST20	$[0.0088, 0.0172]$
ST30	$[0.0069, 0.0124]$
ST40	$[0.0108, 0.0537]$
ST50	$[0.0093, 0.0121]$

Ogni combinazione di soglie trovate sono state utilizzate per effettuare la classificazione tra OK e NOK per ogni singolo giunto saldato. Utilizzando la logica definita precedentemente, si è ricavato l'esito globale delle giunzioni di ogni assem-

blato. Questo a permesso di confrontare gli esiti ricavati dal sistema di computer vision con una precisa combinazione di soglie agli esiti della giuria, e quindi si sono ricavate le tre metriche macro di Recall, Precision e F1-score che si sono descritte nel Capitolo 4.
Iterando il processo descritto precedentemente è stato possibile ricavare questa tripletta di metriche per tutte le 10000 combinazioni e il risultato è visibile in figura 7.12.

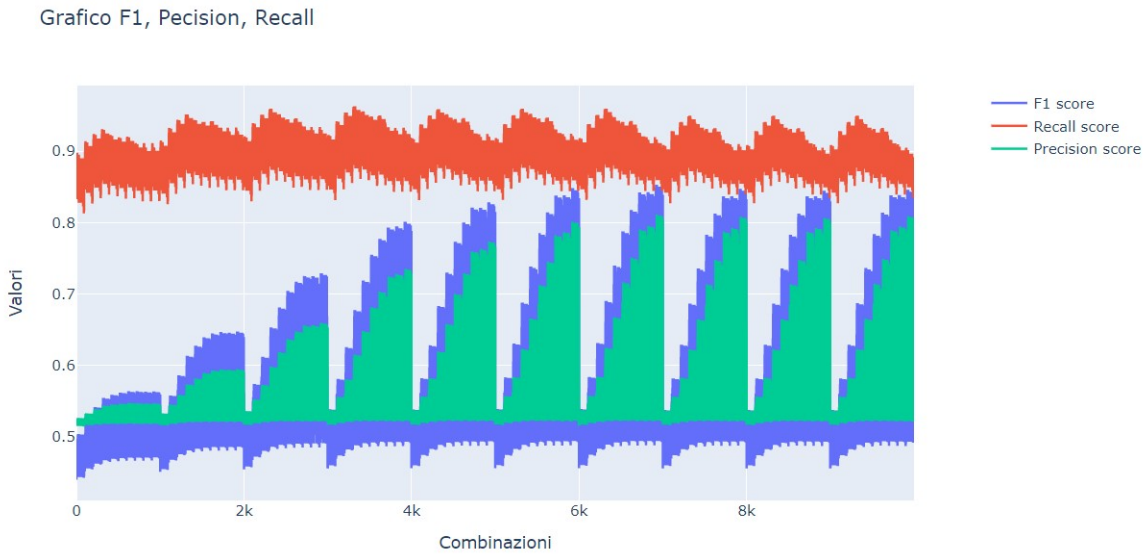


Figura 7.12: Valori di macro F1-score, macro Recall e macro Precision per ogni combinazione delle 4 soglie

Si può notare come tute le curve hanno comportamenti periodici, questo è dato da come sono state costruite le combinazioni, ossia iterando i valori di soglia in maniera periodica in ordine crescente. Infatti si può vedere come si ha un trend crescente della macro Precision e discendente del macro Recall ad aumentare delle soglie, e questo è in accordo con la logica per il quale aumentando i valori delle soglie di controllo delle 4 saldature si avranno un'aumento della precision della classe degli scarti. Per determinare quale siano le migliori combinazioni da prendere in esame si è deciso di utilizzare il grafico in figura 7.13 ossia una rappresentazione tridimensionale dove nell'asse x è rappresentato il macro Recall, nell'asse y è rappresentato la macro Precision e in fine sull'asse z i valori di macro

F1-score. Questo ha permesso di eliminare la componente periodica del grafico in figura 7.12 che rappresenta un variabile di disturbo e di aggregare le combinazioni che portano risulti migliori, ossia con le metriche precedenti più vicine a 1.

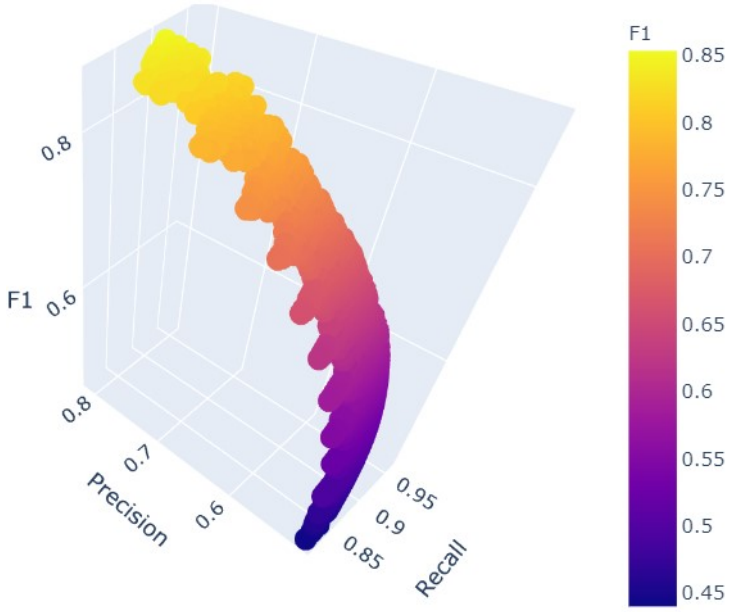


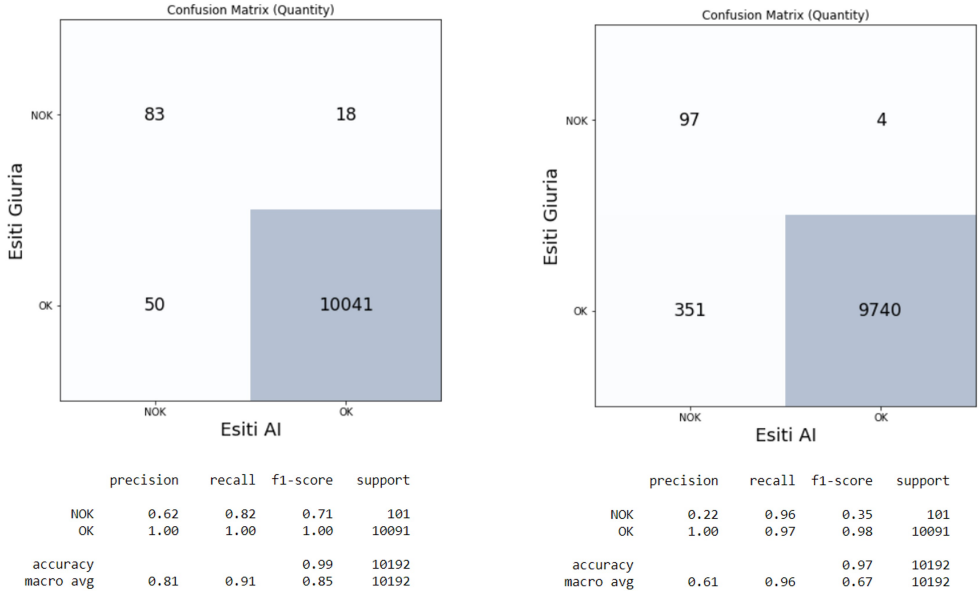
Figura 7.13: Grafico tridimenisonale macro F1 score, macro Recall e macro Precision

Quindi aiutandosi sia con il seguente grafico che con la tabella dove sono salvati i valori delle tre metriche con le rispettive combinazioni, si sono estratte le 2 combinazioni di soglia che massimizzano ognuna una delle metriche tra macro F1-score, macro Recall, macro Precision (quella che massimizza F1, massimizza anche la Precision):

Combinazione soglie (ST20, ST30, ST40, ST50)	**F1 score** (macro)	**Recall** (macro)	**Precision** (macro)
$(0.0144, 0.0124, 0.0156, 0.0121)$	**0.85**	0.91	**0.81**
$(0.0116, 0.0087, 0.0156, 0.0118)$	0.67	**0.96**	0.61

I risultati ottenuti utilizzando queste due soglie, sono riportati in figura 7.14, in cui si hanno due Confusion Matrix con una tabella in cui si riportano il Recall,

Precision e F1 score per entrambe le classi, le loro corrispettive macro e l'accuracy, poco importante per questo caso..



	precision	recall	f1-score	support
NOK	0.62	0.82	0.71	101
OK	1.00	1.00	1.00	10091
accuracy			0.99	10192
macro avg	0.81	0.91	0.85	10192

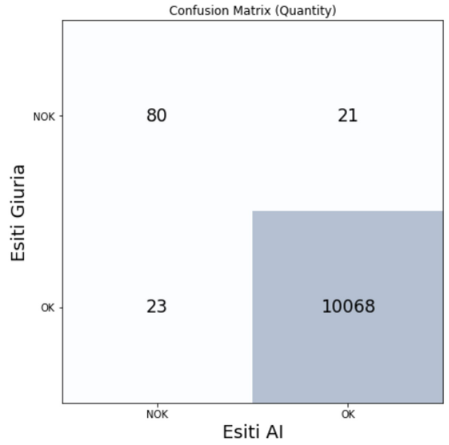
	precision	recall	f1-score	support
NOK	0.22	0.96	0.35	101
OK	1.00	0.97	0.98	10091
accuracy			0.97	10192
macro avg	0.61	0.96	0.67	10192

Figura 7.14: A sinistra sono presenti la confusion matrix e tutte le metriche che si sono ricavate con la combinazione di soglie $(0.0144, 0.0124, 0.0156, 0.0121)$. A destra invece quelle ricavate con la combinazione di soglie $(0.0116, 0.0087, 0.0156, 0.0118)$

Confrontandosi con i dirigenti si è deciso di optare per una soluzione con la prima combinazione in quanto rende il sistema di controllo meno rigido e quindi di scartare complessivamente meno assemblati.
Quindi per concludere, la combinazione di soglie finale scelta con le relative metriche, dopo la rivisitazione manuale delle soglia ST40 e ST50, è la seguente:

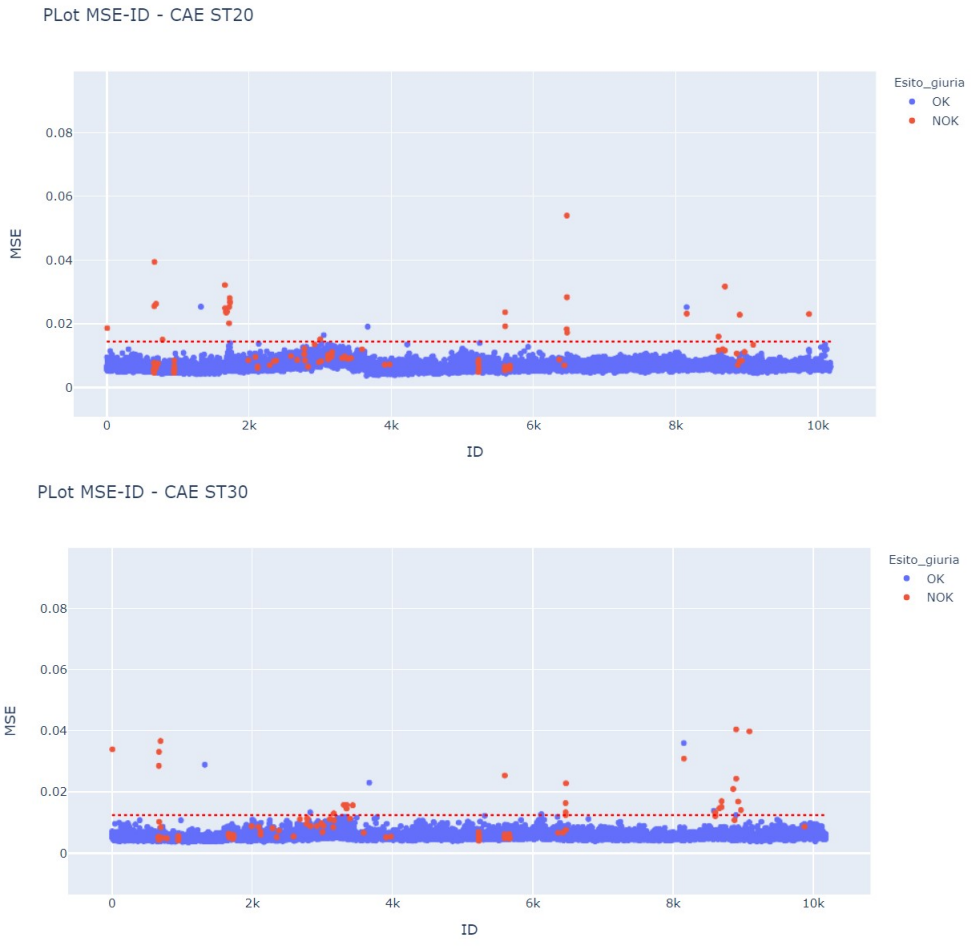
ST20	**ST30**	**ST40**	**ST50**	**F1 score**	**Recall**	**Precision**
0.0144	0.0124	0.013	0.013	**0.89**	**0.89**	**0.89**

In seguito è riportata la confusion matrix con i relativi valori di Recall, Precision e F1 score e i 4 grafici di figura 7.10 con le relative soglie.



	precision	recall	f1-score	support
NOK	0.78	0.79	0.78	101
OK	1.00	1.00	1.00	10091
accuracy			1.00	10192
macro avg	0.89	0.89	0.89	10192

Figura 7.15: Confusion Matrix e tutte le metriche che si sono ricavate con la combinazione di soglie $(0.0144, 0.0124, 0.0130, 0.0130)$



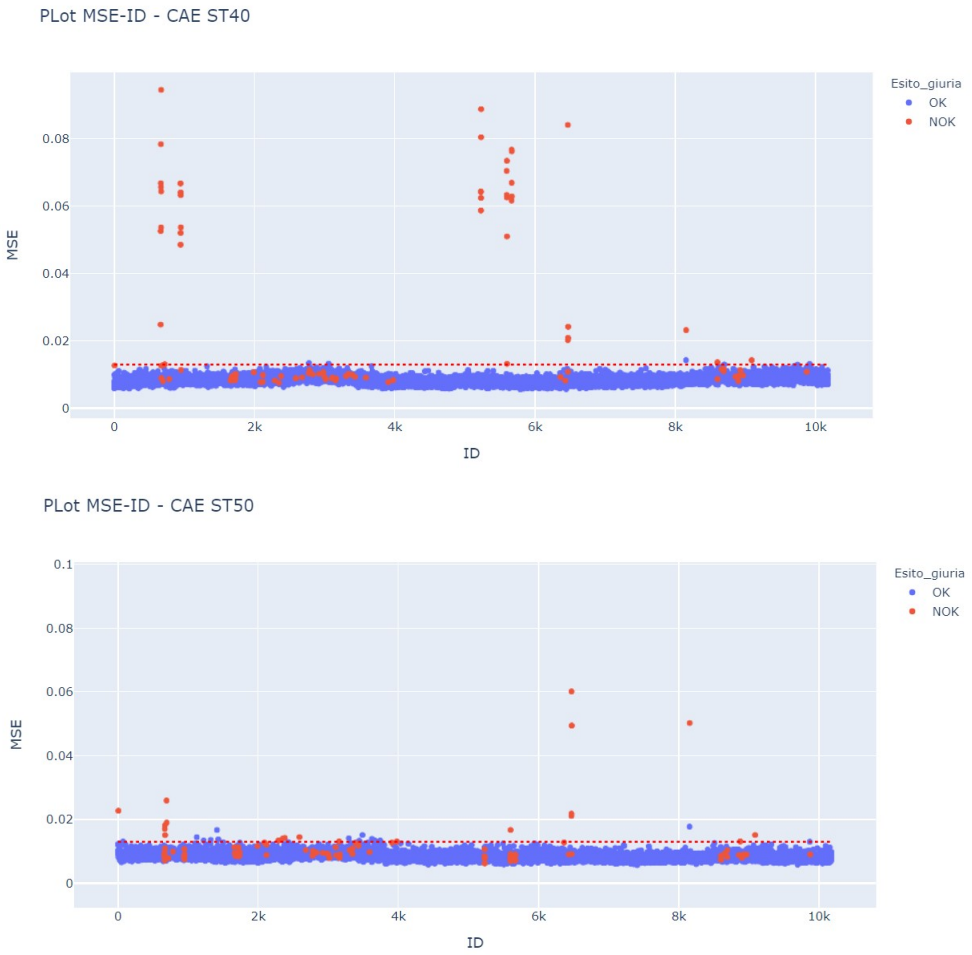
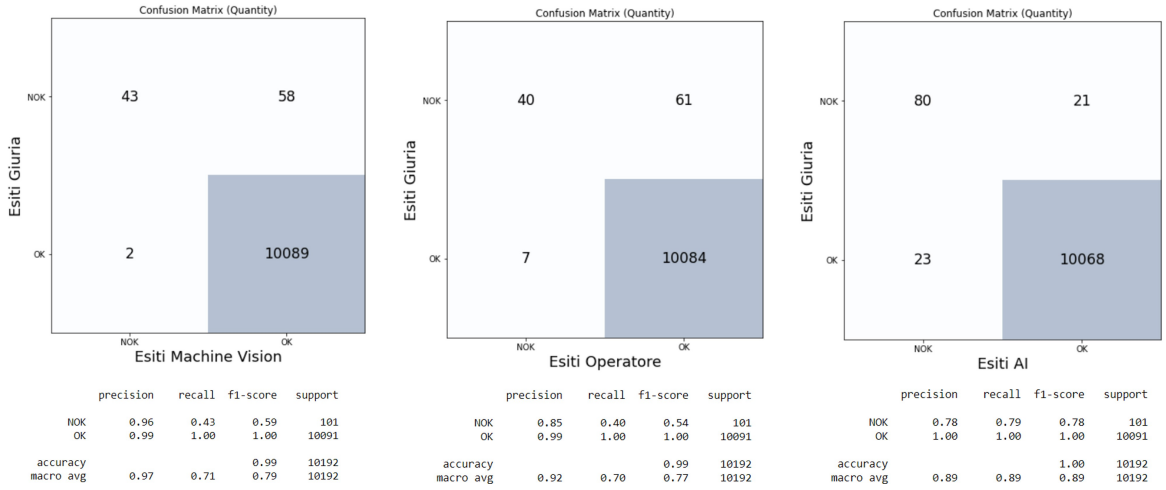


Figura 7.16: Grafici MSE-ID delle stazioni ST20, ST30, ST40, ST50. Le linea rosse tratteggiate in ogni grafico rappresentano la soglia specifica per quella stazione

7.3 Performance rispetto agli altri sistemi di controllo

In questa sezione si confronterà i risultati ottenuti tra i sistemi di controllo qualità attualmente in linea con il sistema sviluppato di computer vision, prendendo sempre come campione le 10192 immagini degli assemblati del giorno 22 e 23 Giugno 2021. In figura 7.17 sono riportate le confusion matrix con i relativi valori di Precision, Recall e F1 score del sistema di machine vision della stazione

ST70, dell'inspezione visiva del operatore e del sistema di computer vision (V5) rispetto alla giuria.



	precision	recall	f1-score	support
NOK	0.96	0.43	0.59	101
OK	0.99	1.00	1.00	10091
accuracy			0.99	10192
macro avg	0.97	0.71	0.79	10192

	precision	recall	f1-score	support
NOK	0.85	0.40	0.54	101
OK	0.99	1.00	1.00	10091
accuracy			0.99	10192
macro avg	0.92	0.70	0.77	10192

	precision	recall	f1-score	support
NOK	0.78	0.79	0.78	101
OK	1.00	1.00	1.00	10091
accuracy			1.00	10192
macro avg	0.89	0.89	0.89	10192

Figura 7.17: Confronto delle performance tra i sistemi attuali di controllo qualità in linea e il sistema di computer vision

Analizzando i risultati precedenti è possibile fare delle precise constatazioni. I sistemi attuali di controllo presi singolarmente sono tarati a ridurre il più possibile i FN comportando un aumento dei FP importante che potrebbero portare a delle non conformità in prossimità del collaudo finale della serratura baule, aumentando così i costi delle inefficienze (il prodotto finito ha un peso a livello di costo molto più alto del semplice carter con saldati due microinteruttori). Parlando di numeri, il sistema di computer vision riesce a bilanciare le due voci in quanto a livello di FP si parla di 21 contro i 61 dell'operatore e i 58 del sistema di machine vision, mentre per i FN la situazione si capovolge in quanto il sistema di computer vision produce 23 falsi scarti rispetto ai 2 del sistema di machine vision e 7 del operatore.

Questo bilanciamento dei FP e FN da parte del sistema sviluppato porta un aumento del macro F1-score rispetto ai sistemi attuali:

/	**Machine vision**	**Operatore**	**Computer vision**
Macro F1-score	0.79	0.77	0.89

Una nota di attenzione merita il valore FP dell'operatore. Molti di quest'ultimi sono difetti palesi come la mancanza di saldatura o mancanza del assemblato nel

posaggio, quindi difetti identificati dal sistema di machine vision. Quasi tutti i FP del sistema di machine vision sono identificati dall'operatore ed infatti se andiamo a vedere quante giunzioni saldate difettate sono passate dal controllo in simultanea dei due sistema precedentemente citati, sono solamente 24 contro i 21 del sistema di computer vision. Quindi se gli attuali sistemi di controllo qualità in linea sono presi singolarmente, il sistema di computer vision è maggiormente performate, mentre se presi insieme, il sistema di computer vision produce maggiori FN (23 contro 8).

7.4 Incertezza di misura e taratura

Come ogni sistema di misura, anche il modello di computer vision sviluppato è soggetto ad una incertezza legata ai disturbi ambientali come vibrazioni e variazioni di luminosità.
In questa trattazione si è valutata l'incertezza seguendo le indicazione presenti nella norma UNI-CEI-ENV 13005 che vede la misura come la media ($\bar{x}$) delle misurazioni mentre l'incertezza come scarto tipo della media che si indica con U_x.
Date N misurazioni valgono le seguenti formule:

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

$$U_x = \frac{S}{\sqrt{N}} = \frac{1}{\sqrt{N}}\sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2}$$

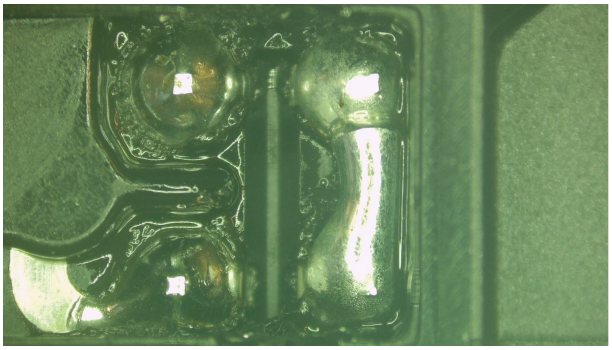
In questo preciso caso per la misura di qualità dei giunti saldati, come detto nel sezione 7.1.5 e presentata a livello matematico nella sezione 3.4.1, si è utilizzata la metrica del MSE che è una media della differenza tra l'intensità dei pixel dell'immagine originale con quella ricostruita dal sistema di anomaly detection. L'unità di misura quindi di questa metrica sarebbe $[Definizione/Risoluzione]$ ossia $[Unit8/pixel]$, ma in questo preciso caso si sono utilizzate le immagini nor-

malizzate per il valore massimo che esso può avere, ossia 255, quindi l'unità di misura sarà $[1/pixel]$.

Per calcolare i seguenti valori, si sono effettuate 40 acquisizioni di immagini di un solo assemblato conforme a distanza di 2 secondi e i risultati delle misurazioni della conformità dei 4 giunti sono visibili in figura 7.18 dove si ha la tabella dei valori di MSE, mentre in figura 7.19 si hanno 4 istogrammi, uno per ogni stazione per visualizzare la loro distribuzione.

Dai dati raccolti si è calcolata la media e lo scarto tipo della media del MSE di ogni stazione e si è utilizzata la rappresentazione della misura descritta nella norma UNI-CEI-ENV 13005:

	Misura completa
ST20	$38.00(22) * 10^{-4} \frac{1}{pixel}$
ST30	$26.30(21) * 10^{-4} \frac{1}{pixel}$
ST40	$40.30(27) * 10^{-4} \frac{1}{pixel}$
ST40	$53.70(28) * 10^{-4} \frac{1}{pixel}$



	ST20	ST30	ST40	ST50		ST20	ST30	ST40	ST50
0	0,00372	0,00251	0,00377	0,00515	20	0,00377	0,00274	0,00416	0,00550
1	0,00349	0,00280	0,00398	0,00509	21	0,00380	0,00260	0,00407	0,00532
2	0,00370	0,00285	0,00437	0,00535	22	0,00389	0,00269	0,00389	0,00536
3	0,00396	0,00254	0,00396	0,00522	23	0,00364	0,00259	0,00400	0,00530
4	0,00379	0,00258	0,00424	0,00537	24	0,00368	0,00265	0,00398	0,00539
5	0,00386	0,00258	0,00387	0,00554	25	0,00395	0,00275	0,00403	0,00561
6	0,00387	0,00263	0,00380	0,00547	26	0,00381	0,00281	0,00406	0,00519
7	0,00386	0,00281	0,00394	0,00570	27	0,00351	0,00246	0,00397	0,00531
8	0,00384	0,00232	0,00411	0,00527	28	0,00388	0,00265	0,00387	0,00530
9	0,00375	0,00285	0,00354	0,00544	29	0,00379	0,00274	0,00413	0,00500
10	0,00371	0,00268	0,00408	0,00546	30	0,00372	0,00245	0,00412	0,00495
11	0,00398	0,00237	0,00431	0,00524	31	0,00389	0,00255	0,00415	0,00544
12	0,00380	0,00281	0,00424	0,00532	32	0,00401	0,00244	0,00415	0,00536
13	0,00370	0,00279	0,00412	0,00538	33	0,00367	0,00262	0,00404	0,00555
14	0,00384	0,00257	0,00406	0,00521	34	0,00378	0,00263	0,00385	0,00559
15	0,00360	0,00252	0,00406	0,00583	35	0,00386	0,00269	0,00403	0,00557
16	0,00388	0,00280	0,00409	0,00531	36	0,00385	0,00250	0,00400	0,00527
17	0,00413	0,00253	0,00423	0,00549	37	0,00393	0,00262	0,00365	0,00535
18	0,00354	0,00264	0,00423	0,00537	38	0,00399	0,00275	0,00393	0,00543
19	0,00389	0,00244	0,00398	0,00534	39	0,00387	0,00263	0,00425	0,00538

Figura 7.18: A sinistra si ha l'assemblato preso di riferimento su cui si sono effettuate le 40 acquisizioni (l'immagine in figura è una tra le 40 acquisite). A destra si ha la tabella dove sono riportate i valori di MSE di ogni singola acquisizione per ogni singolo giunto saldato

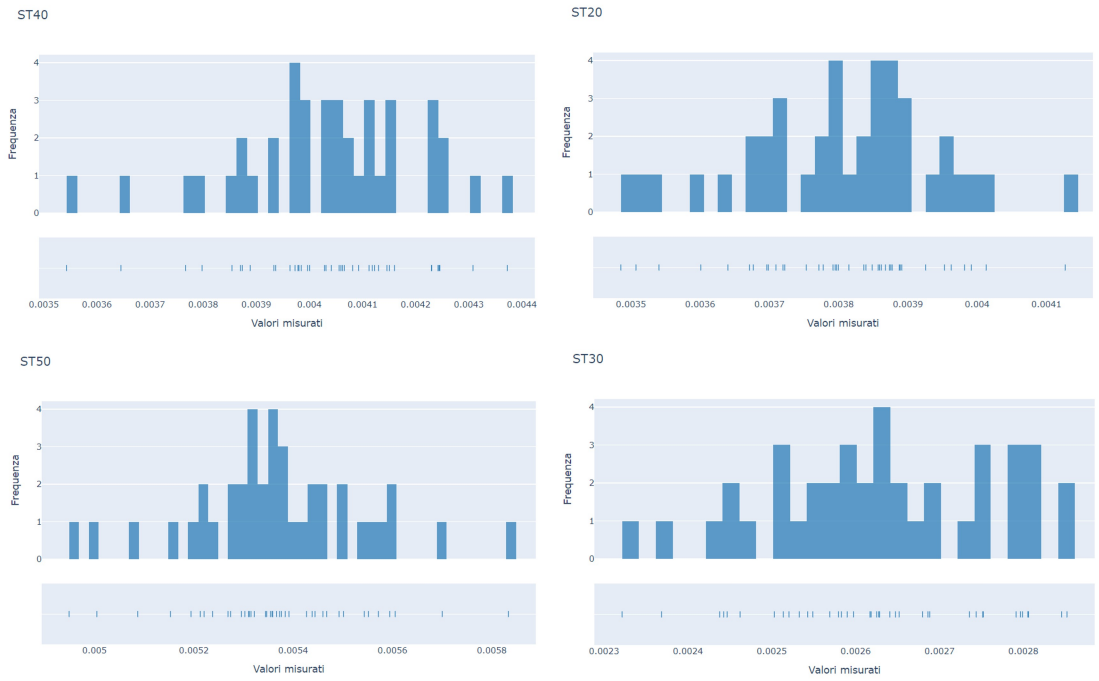


Figura 7.19: Rappresentazione delle distribuzioni delle misurazioni di MSE di ogni singola stazione

Si può notare da questi risultati di incertezza che il sistema di computer vision riesce a restituire la stessa misura di MSE per i giunti saldati a meno di un valore che però difficilmente andrà ad impattare sull'esito della conformità. Quindi la ripetibilità nel breve periodo di questo metodo di misura è alta.
Importante è specificare, che questo test è stato effettuato in condizioni ottimali, ossia dove le uniche fonti di possibile disturbo erano le micro variazioni di luminosità date dalle leggere vibrazioni della tavola rotante. Per le future implementazioni si valuterà anche l'incertezza data dalle micro variazioni di posizione date dalla rotazione tavola, effettuando 40 acquisizioni di un singolo assemblato, ruotando ripetutamente la tavola.

Si è specificato in precedenza che la versione dell'algoritmo descritto è la V5. Queste versioni nascono per tarare il sistema di computer vision alle variazioni dei parametri di saldatura che comportano leggeri cambiamenti a livello di immagine dei giunti o leggere imperfezioni che non vanno a compromettere la conformità

della saldatura.

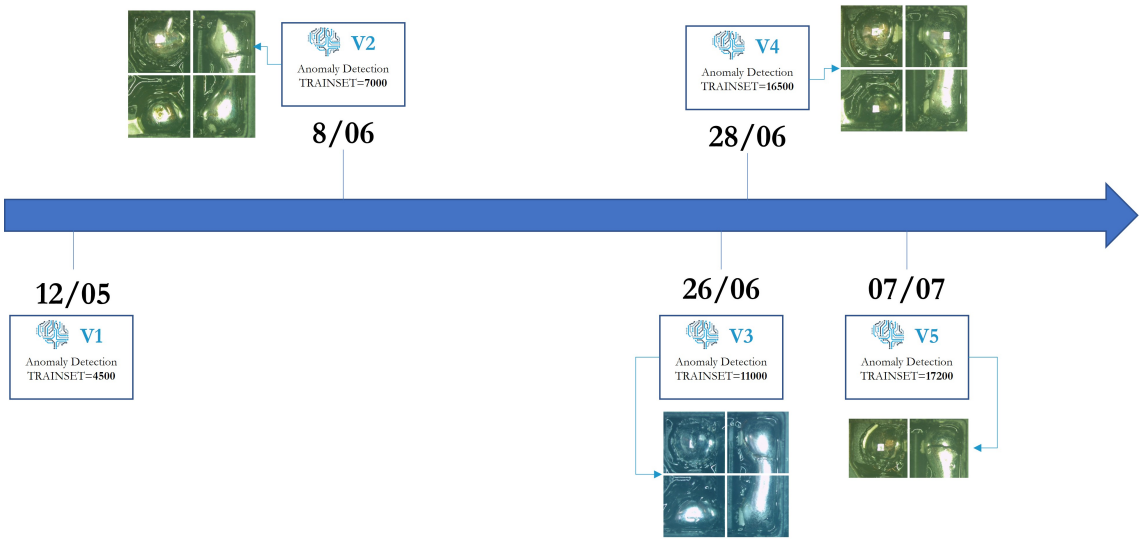


Figura 7.20: Timeline delle versioni dell'algoritmo di anomaly detection

Nella figura 7.20 si posso vedere le tipologie di immagini che si sono aggiunte ad ogni dataset per ogni versione. Nelle V2 si è accettato che il processo di saldatura formasse dei piccoli ghiaccioli nella ST20 e ST40, nella versione V3 si è sono aggiunte immagini di colore più bluastro rispetto alle precedenti dovuto al surriscaldamento del sensore ottico, nella V4 si hanno le immagini dei giunti effettuati in un carter diverso dai precedenti e nella V5 si sono aggiornati esclusivamente gli algoritmi della ST20 e della ST40. Con questo metodo di taratura si va ad aggiungere immagini al dataset di training così da allenare gli algoritmi di anomaly detection a ricostruire immagini di saldatura che precedentemente avrebbero ricostruito in modo sbagliato e quindi restituendo un valore di MSE sopra la soglia di scarto.

Capitolo 8

Conclusioni ed Implementazioni future

Da questo studio e dalla prima implementazione, si è capito che un sistema di computer vision potrebbe essere un perfetto sostituito dei sistemi attualmente in linea per il controllo qualità delle saldature, in particolare del sistema di machine vision. Questo è giustificato dai risultati ottenuti dallo studio sul campione analizzato che possono essere riassunti così:

Controllo	**TN**	**TP**	**FN**	**FP**	**macro F1-score**
Computer Vision	80	10068	23	21	0.89
Machine Vision	43	10089	2	58	0.79

Anche se attualmente il sistema di computer vision produce maggiori falsi scarto (FN) riesce a riconoscere la maggior parte dei difetti rispetto al sistema di machine vision che ha addirittura il numero di FP maggiore rispetto ai TN. Oltre alla miglior performance, il sistema sviluppato ha i seguenti vantaggi rispetto a quello di machine vision:

1. Il sistema di Anomaly Detection, per funzionare ha bisogno esclusivamente delle immagini delle saldature conformi che sono molto più semplici da reperire rispetto ai relativi difetti. Al contrario il tecnico per effettuare un programma di controllo nei sistemi di machine vision ha bisogno dei

possibili difetti per scegliere le aree dove eseguire il controllo. Questo rende il sistema sviluppato più semplice da implementare in nuove applicazioni.

2. I costi relativi all'hardware sono estremamente bassi rispetto ad un sistema di machine vision dedicato.

3. Permette anche di acquisire molti più dati, come le microfermate e quindi sviluppare dei KPI di processo. Questo è possibile conoscendo il deltatime delle immagini raccolte.

Il problema principale di un sistema così pensato è che ogni volta che si hanno alterazioni dei paramenti di processo questo ha bisogno di una taratura, ossia di un nuovo addestramento con le immagini corrispondenti al "nuovo" standard. Se ciò non fosse fatto si alzerebbe il valor medio di errore di ricostruzione (MSE) comportando maggiori falsi scarto. La cosa positiva però è che una volta tarato su un nuovo standard, non perde la capacità di ricostruire immagini di quello vecchio.

I prossimi sviluppi riguarderanno il consolidamento del metodo nel lungo periodo, ossia rendere il sistema funzionante senza ripetere spesso la taratura e quindi avere le tre metriche di valutazione della performance (macro Recall, macro Precision e macro F1-score) più costanti possibili nel tempo.

Sviluppi più sostanziali saranno legati al metodo di classificazione. Si sta studiando un approccio di clusterizzazione per una fascia di MSE intermedia dove il sistema attuale potrebbe fallire (per esempio una saldatura con molto rumore potrebbe avere un MSE pari ad una saldatura con un piccolo cortocircuito). Questo porterebbe ad un raggruppamento per somiglianza utilizzando algoritmi come k-means o tecniche moderne come SPICE[13] (Semantic Pseudo-labeling for Image Clusterin) e fare un classificazione basata oltre che sul MSE anche sull'appartenenza ad un determinato cluster. In un ottica futura questo potrebbe servire a collegare un particolare cluster di saldature a certi paramenti del saldatore robotizzato segnalando la correzione da eseguire per ripristinare lo standard

richiesto, semplificando il lavoro del manutentore, riducendo i tempi di setup e quindi gli scarti per saldature non conformi.

Bibliografia

[1] Guansong Pang, Chunhua Shen, Longbing Cao, Anton van den Hengel. *"Deep Learning for Anomaly Detection: A Review"*. arXiv:2007.02500, 5 December 2020.

[2] Lu Wang, Dongkai Zhang, Jiahao Guo, Yuexing Han. *"Image Anomaly Detection Using Normal Data Only by Latent Space Resampling"*. https://www.mdpi.com/2076-3417/10/23/8660, 3 December 2020.

[3] Charu C. Aggarwal *"Neural Networks and Deep Learning"*. Springer, 2018.

[4] Bruce G Batchelor, Paul F Whelan. *"Intelligent vision systems for industry"*. Springer, 2002.

[5] Rafael C. Gonzalez, Richard E. Woods. *"Digital Image Processing"* Pearson, 2017.

[6] Umme Sara, Morium Akter, Mohammad Shorif Uddin. *"Computer Assisted Image Analysis-Lecture 3-Local Operators"*. JAnders Brun, Uppsala University.

[7] Umme Sara, Morium Akter, Mohammad Shorif Uddin. *"Image Quality Assessment through FSIM, SSIM, MSE and PSNR—A Comparative Study"*. Journal of Computer and Communications, 2019, 7, 8-18.

[8] G.M.Revel. *Slide del corso: Metodi e Strumenti per la diagnostica*. UNIVPM, 2019.

[9] *"Metrics and scoring: quantifying the quality of predictions"*. Scikit Learn: https://scikit-learn.org/stable/modules/model-evaluation.html

[10] *"Measure performance when working with imbalanced data"*. Peltarion: https://peltarion.com/knowledge-center/documentation/evaluation-view/measure-performance-when-working-with-imbalanced-data.

[11] *"JETSON NANO DEVELOPER KIT-User Guide"*. Nvidia, 15 January 2020.

[12] Ernest O. Doebelin. *Strumenti e metodi di misura*. McGraw-Hill, 2008.

[13] Chuang Niu, Ge Wang. *"SPICE: Semantic Pseudo-labeling for Image Clustering"*. arXiv:2103.09382v1, 17 Mar 2021.