UNIVERSITA' POLITECNICA DELLE MARCHE

FACOLTA' DI INGEGNERIA

Corso di Laurea magistrale in Ingegneria Informatica e dell'Automazione

Tecniche di deep learning per il rilevamento di difetti superficiali in materiali compositi

Deep learning techniques for the detection of surface defects in composite materials

Relatore: Chiar.mo

Prof. Dragoni Aldo Franco

Tesi di Laurea di:

Silenzi Andrea

A.A. 2020/2021

Sommario

1 Introduzione

Il presente elaborato finale ha lo scopo di descrivere le attività da me svolte durante il tirocinio effettuato presso l'azienda HP Composites S.p.A. di Ascoli Piceno, per un periodo compreso tra giugno e novembre 2021. Gli argomenti affrontati hanno riguardato lo studio di un sistema di visione artificiale da applicare al rilevamento di difetti superficiali in materiali compositi, in particolare nei casi in cui siano rinforzati con fibra di carbonio. Si tratta di un approccio innovativo per questa tipologia di componenti, che ha come scopo il valutare le potenzialità di un controllo automatizzato della qualità estetica delle superfici in esame, in un contesto in cui lo stesso è effettuato esclusivamente manualmente da operatori specializzati.

Questo studio mirava a sviluppare un metodo efficiente per il riconoscimento delle principali categorie di difetti estetici riscontrati nelle superfici esaminate, basandosi su reti neurali convoluzionali (CNN). Nella soluzione proposta vengono realizzati diversi dataset a partire da immagini acquisite in azienda, successivamente impiegati per addestrare diverse architetture di CNN tra le più utilizzate in letteratura, sfruttando la tecnica del transfer learning.

Le reti VGG-16 e VGG-19 hanno ottenuto i risultati migliori con oltre il 96% di accuratezza per un problema di classificazione a tre classi.

I casi analizzati fanno riferimento alla produzione relativa al periodo in cui si è svolto il tirocinio, ma i risultati possono avere anche una valenza più generale.

1.1 Contesto

Il presente lavoro di tesi nasce da un interesse da parte dell'azienda HP Composites S.p.A., presso cui ho svolto il periodo di tirocinio, di valutare se fosse possibile concepire e realizzare un sistema di visione artificiale per il rilevamento di difetti superficiali in materiali compositi. Come si avrà modo di descrivere in dettaglio nel Capitolo 2, si tratta di un'azienda leader a livello mondiale nella progettazione e produzione di componenti in materiale rinforzato in fibra di carbonio per svariati settori, tra i quali risaltano automotive e motorsport. Il complesso processo industriale, che porta alla realizzazione del prodotto finale, può essere causa dell'insorgenza di numerose tipologie di difetti estetici i quali, sebbene non incidano a livello strutturale, costituiscono un problema di notevole importanza qualora il componente sia venduto "carbon look" e non sia prevista una successiva fase di verniciatura che vada a coprire eventuali difformità superficiali.

La componente estetica è infatti molto rilevante per numerosi clienti di HP Composite, tra i quali possiamo annoverare i più prestigiosi marchi a livello mondiale di auto di lusso o da competizione.
Il controllo della qualità dei singoli pezzi viene tradizionalmente eseguito da operatori esperti che cercano la presenza di qualsiasi incongruenza nella trama delle fibre di carbonio più esterne o di residui di materiali differenti depositati durante le precedenti fasi della lavorazione.
L'assenza di un metodo standardizzato e l'ambiguità di alcune categorie di difetti fa sì che il livello di precisione nell'individuarli sia fortemente soggettivo e dipendente dall'interpretazione dei singoli operatori.
L'introduzione di un eventuale sistema automatizzato, adeguatamente addestrato nel riconoscerli, rappresenterebbe dunque un'assoluta novità e sarebbe concepito per andare ad affiancare tale personale specializzato, fornendo sostegno nelle decisioni più difficilmente interpretabili e fungendo da "arbitro" per eventuali controversie con i clienti finali.
Trattandosi del primo tentativo di automatizzare tale processo e non essendo disponibile del materiale utile nella letteratura scientifica, per la realizzazione di questo primo sistema adatto allo scopo si è deciso di utilizzare tecniche di deep learning, in particolare tramite l'impiego di reti neurali convoluzionali (CNN), le quali rappresentano lo stato dell'arte per la classificazione delle immagini e il rilevamento di oggetti. Essendo considerata un approssimatore di funzione universale, una CNN può essere potenzialmente addestrata per estrarre quasi tutte le features di alto livello dell'immagine di input, comprese quelle relative a difetti di lavorazione. Diverse CNN sono state selezionate come estrattore di feature, confrontandole al fine di classificare le immagini in esame.

1.2 Obiettivi e problematiche iniziali

Lo scopo del presente lavoro è stato quello di fornire un primo sistema che fosse in grado di soddisfare alcune richieste da parte dell'azienda. Si riportano nel seguito i principali obiettivi inizialmente posti, i quali hanno influenzato le successive scelte effettuate per la realizzazione della soluzione proposta:

1. Studio di un sistema di analisi superficiale di materiali compositi per l'individuazione di difetti estetici nella trama della fibra di carbonio.
2. Difetti da rilevare al termine della lavorazione e non sui materiali provenienti dai fornitori e utilizzati a monte nel processo produttivo.

3. Distinzione tra le due macro-categorie, vale a dire difetti recuperabili e difetti non recuperabili.

In aggiunta alle valutazioni precedenti, nelle fasi iniziali dell'attività, ho riscontrato diverse problematiche da tenere in considerazione:

1. Assenza di dataset di interesse nella letteratura scientifica.
2. Geometria complessa e ampia variabilità dei componenti.
3. Riflessi causati dalla natura della superficie.

L'assenza di dataset è stata sicuramente l'ostacolo di maggiore rilevanza in quanto elemento necessario per la fase di addestramento delle CNN. Al fine di risolvere questa problematica, ne sono stati realizzati tre, a partire da immagini di superfici con difetti e prive degli stessi acquisite direttamente sul campo. I successivi due punti hanno invece contribuito a definire il metodo utilizzato per acquisire le immagini.

1.3 Struttura della tesi

L'elaborato è articolato in sette capitoli che trattano le seguenti tematiche:
Nel Capitolo 1 è stato presentato brevemente il contesto applicativo e le principali motivazioni che hanno portato a definire il percorso di tirocinio e tesi.
Nel Capitolo 2 si introduce l'azienda presso cui ho svolto l'attività, descrivendone i mercati in cui opera e fornendo una panoramica generale sui principali processi produttivi che portano alla realizzazione dei materiali compositi. Successivamente verrà affrontato l'argomento oggetto di studio del presente lavoro, vale a dire l'insorgenza di difetti superficiali, conseguenza diretta delle fasi di lavorazione a cui i componenti sono soggetti.
Nel Capitolo 3 viene presentato uno studio sullo stato dell'arte dei sistemi automatizzati per il controllo della qualità di superfici di varia natura. Riportando i passaggi più significativi di articoli presenti in letteratura, si fornirà una panoramica generale delle soluzioni disponibili allo stato attuale, analizzando nello specifico lavori eseguiti sulle superfici che avessero la maggior vicinanza tematica all'argomento in esame, vale a dire metallo, cemento e tessuti. Descrivendone vantaggi e svantaggi per le diverse applicazioni, si valuterà quale possa risultare la scelta migliore da adottare per ottimizzare le prestazioni del sistema oggetto di studio.

Nel Capitolo 4 viene presentato il metodo proposto suddiviso in tre paragrafi che tratteranno i seguenti temi: 1) costruzione dei dataset costituiti da immagini acquisite su componenti reali, descrivendo le valutazioni eseguite per la relativa realizzazione e le principali problematiche riscontrate; 2) panoramica e descrizione delle principali architetture di rete neurale convoluzionale utilizzate per svolgere il compito di classificazione; 3) breve cenno relativo alle tecnologie che hanno permesso la scrittura e l'esecuzione dei codici.

Nel Capitolo 5 vengono descritti nel dettaglio i vari esperimenti svolti, descrivendo la pipeline di preprocessing a cui sono sottoposte le immagini prima di essere date in ingresso alle reti, i principali criteri di valutazione e i risultati ottenuti per i differenti dataset testati.

Nel Capitolo 6 viene eseguita una valutazione delle performance dei diversi algoritmi e un'analisi dei risultati migliori, e vengono proposti dei possibili sviluppi futuri, con l'obiettivo di fornire utili spunti che possano aiutare a migliorare quanto realizzato e ad estenderne le funzionalità.

Infine, nel Capitolo 7 verranno tratte le conclusioni e si farà una considerazione generale del lavoro svolto.

2 Scenario

Il presente lavoro di tesi è stato reso possibile grazie alla collaborazione dell'azienda HP Composites S.p.A. di Ascoli Piceno, presso cui ho svolto il periodo di tirocinio curricolare previsto dal piano di studi del corso di laurea magistrale in Ingegneria Informatica e dell'Automazione.
Di seguito viene fornita una descrizione dell'azienda e delle principali attività che svolge al proprio interno, delle tecnologie più rilevanti di cui fanno uso per la realizzazione dei propri prodotti e di alcune conseguenze negative che tali lavorazioni comportano.

2.1 HP Composites S.p.A.

HP Composites [1] è un'azienda leader a livello mondiale nella progettazione e produzione di componenti in materiale rinforzato in fibra di carbonio per i seguenti settori:

Automotive

Si tratta del settore leader, in cui sono richieste dedizione, ripetibilità e tracciabilità del processo, in modo tale da soddisfare qualsiasi esigenza del cliente. La capacità produttiva di HP continua a crescere assieme allo sviluppo di processi produttivi innovativi che permettono la riduzione di tempi e costi, senza però compromettere la qualità del prodotto finale.

Motorsport

I componenti richiesti da questo settore sono estremamente performanti e hanno un'alta reattività. Le competenze interne all'azienda e il reparto R&D offrono continuamente nuove soluzioni che possano aumentare le proprietà meccaniche dei prodotti.

Aereonautico

L'industria aereonautica è stata la prima ad aver fatto uso di materiali compositi. In questo settore è infatti richiesto il massimo della qualità del prodotto e del controllo dei processi. Allo stato attuale, l'azienda è coinvolta nella produzione degli interni, nella fornitura di stampi e attrezzature per compagnie aeree.

Industriale

Il settore industriale offre numerose opportunità di impiego. Ad esempio, la possibilità di alleggerire il carico di un componente meccanico consente di risparmiare energia, aumentare l'operatività e migliorare il ciclo di vita del prodotto a vantaggio della riduzione del tempo e dei costi.

Navale

In questo settore si è registrato un costante aumento nell'utilizzo dei materiali compositi data la loro leggerezza, estetica e funzionalità. HP Composites è partner di alcune case produttrici di yachts nel fornire diverse componenti strutturali.

Design

Grazie al successo ottenuto nei vari campi applicativi di cui sopra, l'azienda ha iniziato a produrre mobili di alto livello. Il settore del lusso e del design in generale richiede di bilanciare la natura strutturale dei componenti, mantenendo un perfetto risultato estetico prestando attenzione al dettaglio [2].

Allo stato attuale HP consta di cinque stabilimenti produttivi che coprono un'area complessiva di 22.000 mq. Dal 2010, anno di nascita, ad oggi ha aumentato il numero di risorse umane presenti all'interno dell'azienda passando da 40 a 600 dipendenti durante i periodi di picco produttivo.

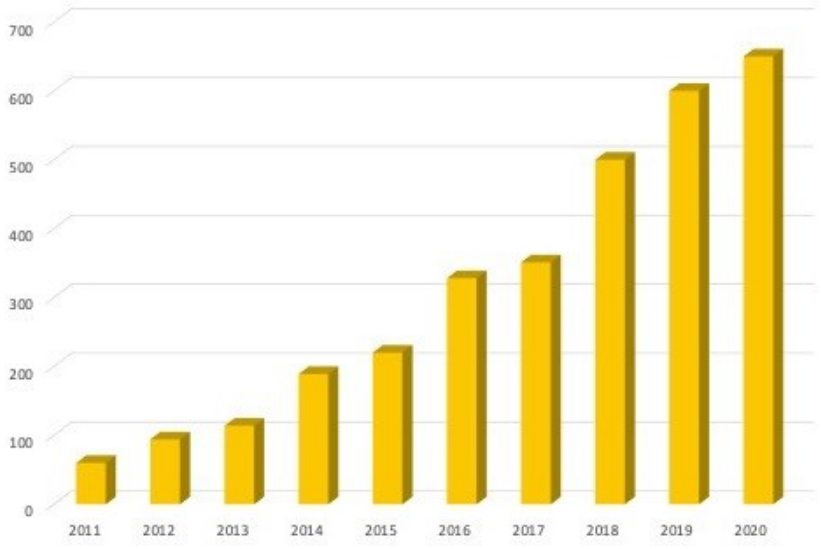

Figura 1: Crescita del numero dei lavoratori di HP Composites [3]

2.2 Materiali compositi

Un materiale composito può essere definito come un sistema costituito da due o più fasi, le cui proprietà e prestazioni sono progettate in modo tale da essere superiori a quelle dei materiali costituenti che agiscono indipendentemente. Normalmente una delle due fasi è discontinua, più rigida e resistente ed è chiamata rinforzo, mentre l'altra è continua, meno rigida e più debole ed è chiamata matrice. A volte è presente una fase ulteriore tra le due precedenti, chiamata interfase.
Le fasi hanno ruoli differenti in base al tipo di utilizzo che viene fatto del materiale composito di cui fanno parte. Ad esempio, nel caso in cui le prestazioni richieste siano di basso o medio livello, il rinforzo, usualmente costituito da fibre corte o particelle, fornisce un irrigidimento ma rinforza il materiale solo a livello locale, mentre la matrice è il costituente principale per reggere i carichi e definire le proprietà meccaniche. Nel caso di materiali compositi ad alte prestazioni, invece, il rinforzo è normalmente costituito da fibra continua e determina la rigidezza e resistenza lungo la direzione della fibra, mentre la matrice fornisce protezione, sostegno e il trasferimento degli sforzi locali da una fibra all'altra [4].

2.2.1 Fibra di carbonio

Grazie alle loro dimensioni limitate, le fibre presentano caratteristiche strutturali uniche che, unite alle proprietà intrinseche dei materiali costitutivi, assicura ad esse resistenza meccanica elevata, alto modulo elastico, basso peso specifico e comportamento elastico lineare fino alla rottura.
Le fibre più importanti nel mondo dei compositi possono essere di vetro, organiche, minerali o di carbonio. Queste ultime, date le elevate proprietà meccaniche dovute alla particolare struttura cristallina della grafite, negli anni hanno in larga parte sostituito le più economiche fibre di vetro in tutti quei campi in cui sono richieste, oltre ad un basso peso, un'alta rigidità o una notevole stabilità dimensionale al variare della temperatura [4].
Tali fibre, oltre alle proprietà brevemente elencate che contribuiscono nel definire il comportamento strutturale dei materiali compositi di cui sono costituenti, sono anche alla base della resa estetica del prodotto finale, nei casi in cui sia prevista la commercializzazione dei componenti con la struttura del tessuto a vista, ed è in questo contesto che si colloca il presente lavoro.

2.2.2 Processo produttivo

Le tecnologie di fabbricazione utilizzabili per la realizzazione di elementi in materiali compositi sono numerose e variano a seconda della forma, dimensione e proprietà richieste al pezzo finito.
Di seguito verranno brevemente descritte le principali tecniche adottate all'interno dell'azienda.

Autoclave

Utilizzata per la produzione di parti complesse come le monoscocche o pezzi destinati al settore aereospaziale. Pressione e temperatura interna sono controllate così da realizzare componenti in maniera differente, in base alla dimensione e al materiale utilizzato. Attraverso questa tecnologia vengono prodotte parti in materiale composito dove le resine sono epossidiche e il rinforzo è in fibra di carbonio, kevlar e/o altre fibre. Risulta essere adatta anche per la produzione di parti strutturali come carrozzeria e componenti in carbon look.

Press moulding

Questo sistema permette di ottenere un'elevata qualità e uniformità estetica con tempi ciclo brevi. La ripetibilità consente di produrre in serie parti strutturali e in carbon look. Tramite questa tecnologia è possibile produrre pezzi di piccole dimensioni e di media complessità.

Compression moulding

Il ciclo di polimerizzazione è svolto sotto pressa all'interno di stampi riscaldati. Normalmente si utilizza fibra corta e si adatta alla costruzione di piccole parti.

Forming pre-preg

Tecnologia sviluppata nel settore aereonautico per industrializzare la fase di laminazione manuale. Consiste nella "formatura a caldo" di pacchetti di pre-preg (pre-impregnati) da stampare poi in pressa o in autoclave. La formatura avviene grazie all'applicazione di calore e vuoto su un pacchetto di pre-preg posto su stampo di formatura.

Air press moulding

Tecnologia sviluppata da HP Composites, nel tentativo di creare un processo produttivo che si posizionasse tra la produzione in autoclave e quella in pressa, sfruttando i vantaggi di entrambe. Questa tecnologia è ideale per la produzione di componenti di grandi dimensioni. Il suo utilizzo consente di ottenere un'elevata produttività, mantenendo l'equilibrio tra costi e prestazioni.

RTM

Tecnologia tradizionale che consiste nell'iniezione di resina in pressione attraverso preforme di fibra secca contenute all'interno di stampi rigidi chiusi, consentendo di ottenere un limitato rapporto di fibra/resina. Si adatta a geometrie semplici, ad elevato spessore e preferibilmente dalle dimensioni non eccessive. Normalmente gli stampi (metallici) sono riscaldati e contrastano le forze di iniezione.

VARTM

Simile alla tecnologia RTM, con la differenza che le pressioni di iniezione sono molto basse e assistite dal vuoto. Non sono necessari controstampi metallici e pressa. Di contro si ha che le performance raggiungibili sono molto limitate. HP ha sviluppato una variante del VARTM e dell'infusione che consente di iniettare resine più reattive e di ridurre drasticamente i contenuti di vuoto.

Space frame technology

Tecnologia che consente di ottenere un prodotto finito ad elevate performance, costituito da diverse parti, anche costruite con tecnologie differenti. Le parti sono assemblate tra di loro garantendone la continuità strutturale grazie a riporti locali di pre-preg, il tutto processato in autoclave (a caldo e sotto pressione) a garanzia delle prestazioni finali.

C-RTM

Tecnologia sviluppata da HP Composites che combina i vantaggi della produzione in autoclave (assenza di vuoto, tempi ciclo brevi, semplicità nell'apportare una modifica dello spessore) e quella dell'RTM (basso costo dei materiali, alto tasso di produzione) [5].

Nella figura sottostante è mostrata una schematizzazione delle fasi di lavorazione che permettono di ottenere un prodotto finito. Nello specifico, è possibile osservare come, a partire da bobine di fibre intrecciate tra loro, si realizzi il materiale composito, con le fibre di rinforzo tessute e annegate in una matrice di resina. Questa combinazione costituisce i pre-preg, i quali vengono successivamente tagliati e uniti a degli stampi nella camera di laminazione, procedura che avviene all'interno di cleanroom in cui gli operatori indossano indumenti appropriati per mantenere al minimo il livello di contaminazione con l'esterno.

Alla conclusione della laminazione è di fondamentale importanza l'inserimento dello stampo in appositi sacchi da vuoto, la cui funzione è di permettere una corretta polimerizzazione in autoclave. Nel caso in cui la chiusura del sacco non fosse ermetica, verrebbe meno la depressione e il processo di polimerizzazione ne risulterebbe compromesso, a volte solo dal punto di vista estetico, altre volte minandone anche la resistenza meccanica, comportando una non idoneità alla vendita. I sacchi da vuoto sono realizzati con materiale plastico, la cui resistenza termica deve essere molto elevata per sostenere i programmi di temperatura necessari alla polimerizzazione. Altra caratteristica di fondamentale importanza è rappresentata da una grande elasticità, la cui funzione è di far aderire l'intera superficie del manufatto al sacco, consentendo un'omogenea distribuzione del vuoto [6].

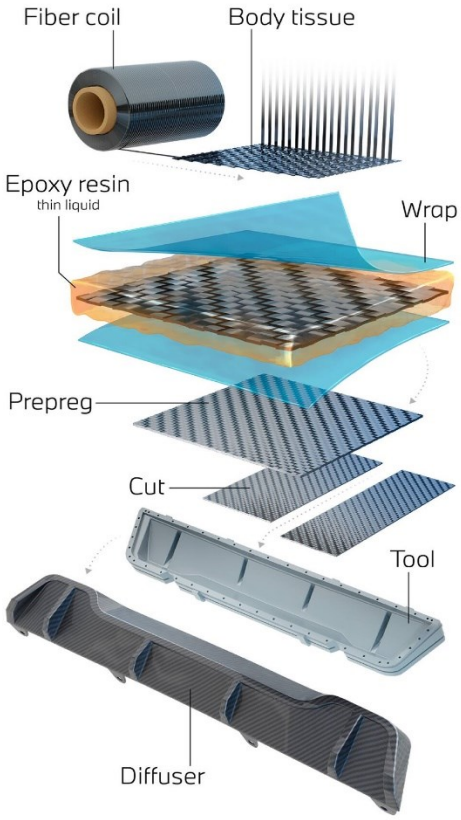

Figura 2: Fasi produttive di un materiale composito [7]

2.3 Difetti estetici

Le fasi di lavorazione precedentemente descritte possono in alcuni casi generare delle difettosità superficiali. Non è infrequente, infatti, una volta estratto il componente dallo stampo, notare delle piccole o grandi incongruenze nella trama della fibra di carbonio a vista. Tali tipologie di difetti non influenzano in alcun modo le caratteristiche meccaniche del prodotto finale, ma sono da evitarsi solo per motivi di gradevolezza estetica. Prima di poter essere considerato idoneo alla vendita, ogni componente viene esaminato da personale specializzato con il compito di individuare tali difettosità e, qualora fossero presenti, prendere in considerazione la possibilità di recuperare il pezzo cercando di porvi rimedio o scartarlo. I difetti riscontrabili possono quindi essere raggruppati in due macro-categorie, vale a dire difetti recuperabili e difetti non recuperabili. Nel seguito della trattazione verranno approfonditi questi due aspetti, introducendo i difetti più comuni riscontrati durante il periodo di tirocinio.

2.3.1 Difetti estetici recuperabili

Rappresentano la categoria di difetti riscontrata con maggiore frequenza e non sono di grande impatto in quanto facilmente eliminabili nel ciclo di rifinitura. Solitamente sono trattati con la deposizione di resina trasparente, la cui funzione è di uniformare la superficie, e successiva levigatura con carta abrasiva. A seconda del tipo di resina utilizzata, la polimerizzazione può essere effettuata all'aria o tramite l'impiego di raggi UV. Un ulteriore metodo usato per correggere il problema laddove le parti soggette siano poco visibili, consiste nella stesura di collante strutturale nero. La correzione di questi difetti è riscontrabile nei casi in cui il prodotto finale sia venduto con la caratteristica trama delle fibre a vista, e non si pone in pezzi per cui è prevista una successiva fase di verniciatura che, coprendo l'eventuale difetto, costituirebbe essa stessa il rimedio.

2.3.1.1 Porosità isolate

Le porosità rappresentano una delle tipologie di difetto più comune, rinvenute in numerosi componenti una volta estratti dallo stampo. La loro formazione avviene principalmente durante la polimerizzazione in autoclave ed è spesso causata da parametri errati o non ottimali come la durata, la temperatura o la perdita di depressione del sacco da vuoto. Possono essere presenti anche al di sotto della superficie sotto forma di vuoti all'interno della matrice, e in tal caso costituiscono un problema assai più di rilievo in quanto possibile causa di cedimenti strutturali del componente.

In questa trattazione verranno prese in considerazione le sole porosità superficiali, la cui presenza è da evitare solo per un fine puramente estetico e non di tenuta strutturale. Quando presenti sulla parte più esterna del composito, assumono l'aspetto di fori, che possono essere presenti anche in gran numero ma, purché siano di piccole dimensioni, non vadano troppo in profondità e non interessino un'area sufficientemente estesa, rappresentano la problematica che desta le minori preoccupazioni, in quanto facilmente risolvibile con la deposizione di resina trasparente e successiva levigatura. Rientrano inoltre tra le difettosità più semplici da individuare in quanto la particolare natura della superficie, interessata da un adeguato sistema di illuminazione, consente di farli risaltare rispetto allo sfondo.

Nelle immagini sottostanti vengono presentati alcuni esempi di porosità isolate. Si noti come risaltino in maniera differente in base alla diversa illuminazione del componente, rimanendo tuttavia facilmente distinguibili rispetto alle aree integre circostanti.

Figura 3: Esempi di porosità isolate

2.3.1.2 Infiltrazioni

Altri difetti rinvenibili più di rado e che non comportano lo scarto del pezzo sono rappresentati dalle infiltrazioni di corpi estranei al di sopra della superficie. Infatti, in alcuni casi sono state riscontrate piccole infiltrazioni di alluminio e polietilene, ma anche in questo caso il problema può essere facilmente risolto andando ad applicare una piccola leva per staccare tali depositi, facendo attenzione a non danneggiare l'area sottostante. Il recupero di tale difetto è reso possibile in quanto risulta in rilievo rispetto alla superficie. Le immagini seguenti chiariscono meglio quanto esposto, nelle quali è possibile notare anche la presenza di piccole porosità.

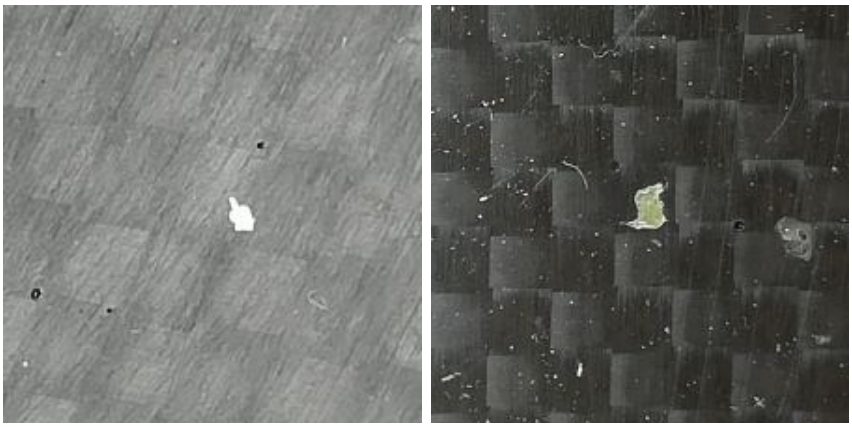

Figura 4: Esempi di infiltrazione di alluminio e polietilene

2.3.2 Difetti estetici non recuperabili

Questa categoria di difetti include tutte le casistiche che se presenti comportano lo scarto del pezzo, a meno di copertura con vernice. La loro maggiore importanza è data dall'impossibilità di eliminarli in una fase successiva alla lavorazione.

La causa dell'insorgenza di tali difetti è spesso imputabile ad una errata operazione manuale effettuata nella fase di laminazione o applicazione del sacco da vuoto.

Durante il periodo di tirocinio ho avuto modo di osservarne differenti tipologie, di cui verrà fornita una descrizione nel seguito.

2.3.2.1 Discontinuità nella trama

Sono inclusi in questa categoria tutti i casi in cui la caratteristica trama dei fasci di fibre di carbonio tra loro intrecciati risulta alterata. I possibili scenari che possono presentarsi sono numerosi, tuttavia i più comuni hanno riguardato una interruzione della continuità della trama o incongruenze visive in genere. La sede di maggiore insorgenza di questi difetti si è riscontrata in prossimità dei bordi dei manufatti, in presenza quindi di curvature geometriche. Generalmente la loro formazione è determinata da un'errata deposizione delle pelli che sovrapposte formano il componente o da una scarsa adesione delle fibre allo stampo. Sebbene spesso si trovino in prossimità di curvature, si tratta dei più difficili da rilevare in quanto non risultano in rilievo rispetto la superficie più esterna ma sono parte della stessa. Inoltre, mentre in alcuni casi la differenza tra trama integra e distorta è evidente, in altri può essere rilevata solo da personale specializzato, che tuttavia non può garantire un controllo standardizzato su ogni componente. Alcuni esempi a riguardo sono riportati di seguito.

Figura 5: Esempi di discontinuità della trama

In altri casi si assiste ad uno scollamento dei vari strati sovrapposti che costituiscono i componenti, dovuto ad una mancata compattazione superficiale, come mostrato in Figura 6.

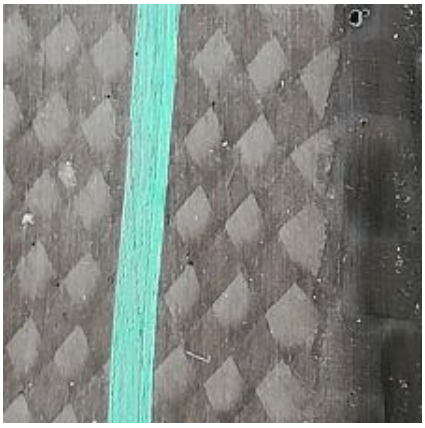

Figura 6: Mancata compattazione superficiale

Vi sono inoltre casi in cui l'interruzione della continuità della trama non è dovuta a deformazioni, bensì ad una errata fase di assemblaggio in componenti costituiti da più parti, come mostrato nelle immagini seguenti.

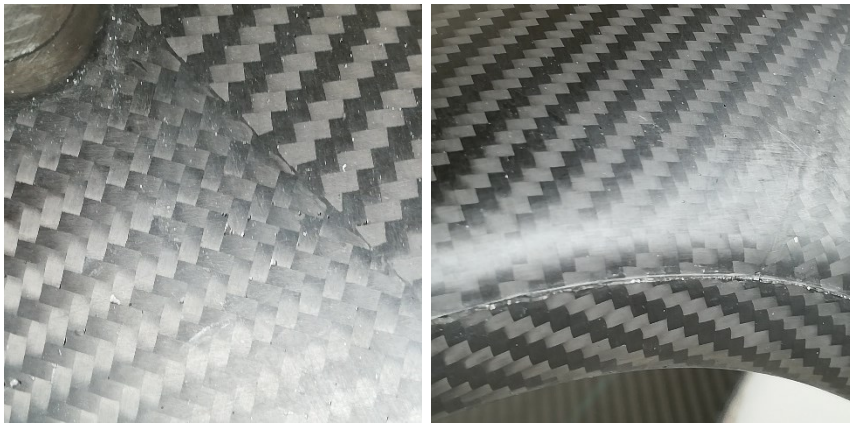

Figura 7: Interruzione continuità del motivo

2.3.2.2 Accumulo di resina

Un altro difetto a volte riscontrato è riportato in Figura 8, in cui si possono osservare delle macchie presenti al di sopra e nel mezzo della trama. Si tratta di accumuli di resina spesso causati da una imperfetta calibratura degli spazi che intercorrono tra i due stampi e il mandrino di silicone interposto tra essi. A seconda del tipo di resina utilizzata come matrice e al variare dell'illuminazione possono assumere aspetti differenti, presentandosi il più delle volte con colori che vanno dal giallo al marrone/violaceo o nero. Nella maggior parte dei casi, i più evidenti sono presenti in prossimità di alcuni spigoli negativi del pezzo e rappresentano la principale causa di scarto dello stesso, ma ad uno sguardo più attento si può notare come l'intera superficie sia interessata, sebbene in forma minore. Vale la pena sottolineare come in alcune situazioni meno gravi il componente possa essere recuperato cancellando l'accumulo con la deposizione di resina trasparente e successiva levigatura.

Figura 8: Esempi di accumulo di resina

2.3.2.3 Eccessiva porosità

Viene infine riportato un problema già incontrato in precedenza in una versione di minore impatto, vale a dire la presenza di porosità. A differenza del caso precedente in cui erano caratterizzate dall'essere isolate e di piccole dimensioni, nel seguente sono presenti in gran numero e interessano gran parte della superficie, andando anche maggiormente in profondità, non essendo recuperabili in alcun modo e comportando inevitabilmente lo scarto del pezzo. Si tratta forse del tipo di difetto più semplice da individuare in quanto immediatamente riconoscibile e molto esteso, spesso accompagnato con una distribuzione non uniforme della resina. Anche in questo caso la causa è da rinvenire in un errato uso che viene fatto del sacco da vuoto, che deve essere correttamente posto in modo che non si formino pieghe o punti di non adesione che potrebbero influenzare la resa

estetica del pezzo, creando zone di disomogeneità nella distribuzione della resina oppure la riproduzione della piega sulla superficie del pezzo [6]. In Figura 9 sono riportati gli effetti della perdita di depressione: venendo meno l'omogeneità della pressione assicurata dal sacco da vuoto, non è stato possibile completare il processo di polimerizzazione e, ove esso è stato completato, la distribuzione della resina è risultata non uniforme.

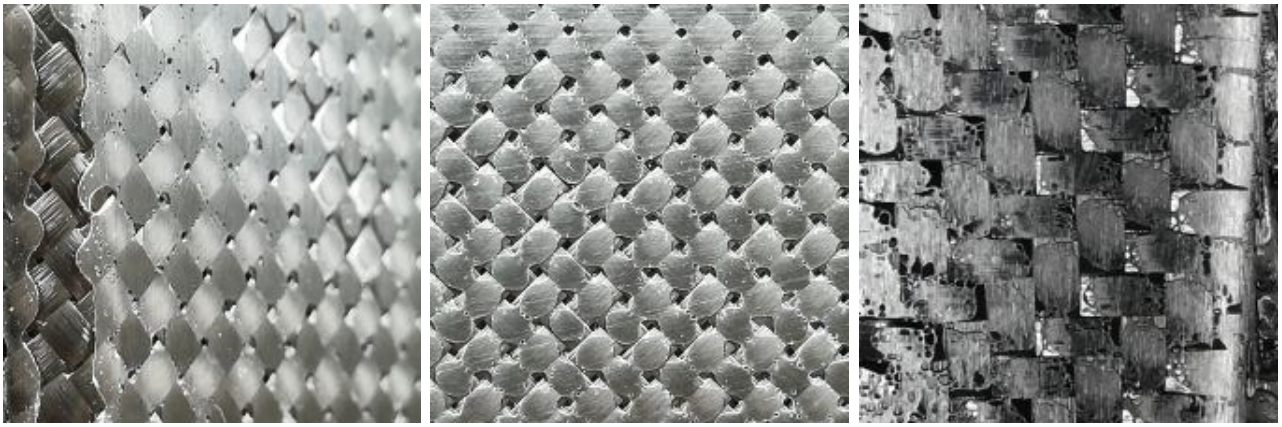

Figura 9: Eccessiva porosità e disomogeneità di distribuzione della resina

3 Stato dell’arte

La realizzazione del progetto è dipesa da diverse considerazioni relative alle tecnologie da impiegare tra le varie possibili. Una prima distinzione può essere fatta considerando le due grandi categorie che racchiudono differenti metodi con cui viene affrontato il problema, vale a dire le tecniche tradizionali di computer vision o le più recenti e dalle sempre maggiori potenzialità di deep learning. Di seguito viene proposto uno studio relativo agli scenari possibili basandosi su ciò che è presente in letteratura, relativamente al controllo della qualità in superfici di varia natura che presentino le caratteristiche maggiormente simili ai materiali compositi. Lo scopo di tale ricerca è di individuare le tecniche che più si adattano al contesto in esame per valutare la potenzialità di approcci di questo tipo, non essendo stato trovato alcuno studio che trattasse direttamente l’argomento. A riguardo, vale la pena sottolineare come la maggior parte dei lavori presenti in letteratura e dei reali sistemi intelligenti utilizzati per il controllo della qualità siano applicati a superfici piane o tessuti che scorrono su di un nastro trasportatore. La difficoltà del tema in esame deriva proprio dal tipo di richiesta avanzata dall’azienda ospitante il tirocinio, ovvero di studiare un sistema applicabile su componenti a tre dimensioni, di grandezza diversa e dalle geometrie complesse.

3.1 Panoramica

La tematica riguardante il controllo qualità automatizzato per il rilevamento di difetti superficiali è da tempo oggetto di studio e alla base di numerose ricerche in ambito scientifico, volte ad ottimizzare tali fasi e porre rimedio alle inefficienze nei processi industriali, costose in termini di tempo, denaro e soddisfazione del consumatore. L'economia globale e le pressioni a cui sono sottoposte hanno gradualmente portato le aziende a ricercare una sempre crescente efficienza produttiva finalizzata ad una maggiore competitività nel mercato. Di conseguenza, è in costante aumento la domanda per sistemi di ispezione visiva intelligenti che garantiscano un’alta qualità dei prodotti nelle linee di produzione. I materiali per i quali trovano impiego questi sistemi sono numerosi e, nonostante siano disponibili in molte forme, è riscontrabile una notevole somiglianza nei requisiti di automazione richiesti. In base al tipo di materiale possono essere distinte due categorie, dalla presenza di una superficie uniforme o con delle texture [8].

La prima categoria è associata a materiali prodotti sotto forma di rotoli continui uniformi come metalli, film, carta, etc. Il rilevamento di difetti in questi materiali normalmente si basa su identificazione di regioni che differiscono da uno sfondo uniforme.
La seconda categoria è associata a materiali strutturali come tessuti, ceramiche, plastiche, etc. La percezione di ciò che costituisce un difetto strutturato varia da individuo a individuo e spesso la stessa persona può avere una sensibilità diversa di volta in volta [9]. La caratterizzazione dei difetti nei materiali con texture non è generalmente definita in maniera chiara. Pertanto, la loro ispezione visiva consiste nel classificarli in base alle caratteristiche di texture complessive come l'isotropia del materiale, l'omogeneità e la grossolanità della texture.
I materiali con texture possono ulteriormente essere suddivisi in uniformi, a trame casuali o con motivi. L'ispezione di difetti reali nei tessuti, categoria nella quale può essere inclusa la fibra di carbonio, è particolarmente impegnativa a causa di variazioni aleatorie di scala, allungamento e inclinazione della trama del tessuto, prevalentemente a causa dell'ambiente e della natura del processo di tessitura coinvolto.

3.2 Strumentazione

Un sistema di rilevamento difetti utilizzabile a livello industriale non può fare a meno di un adeguato insieme di componenti sincronizzati tra loro e che siano in grado di operare alla stessa velocità con cui vengono prodotti i pezzi lungo la linea di produzione. In Figura 10 è riportata l'architettura generale di un sistema di ispezione per linee di produzione in cui vi sia la presenza di un nastro trasportatore, come ad esempio per l'industria tessile.
Ajay Kumar [8] descrive gli elementi caratteristici di un sistema di ispezione per l'industria tessile. Esso è costituito da un banco di telecamere disposte in parallelo rispetto al rotolo continuo da scansionare, un sistema per la ricezione e l'elaborazione dei dati, il frame grabber, un sistema di illuminazione e le interfacce elettrica e meccanica di supporto per la macchina di ispezione. Spesso il sistema di ispezione impiega un grande parallelismo nell'acquisizione delle immagini tramite un algoritmo che riduce il flusso di dati alla sola regione di interesse. Risulta chiaro come un sistema di questo tipo sia costituito da alcuni elementi chiave, di cui verranno descritte le caratteristiche di seguito:

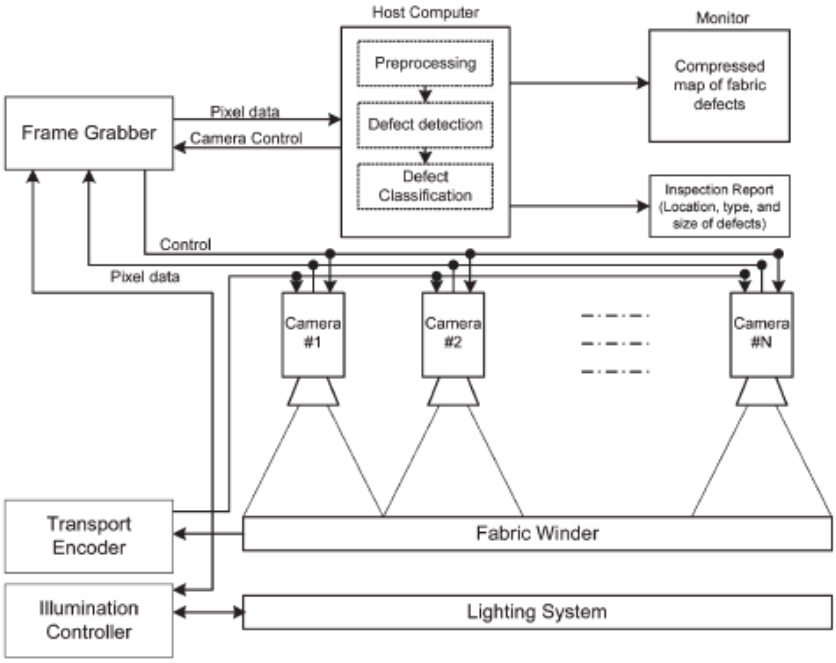

Figura 10: Esempio di architettura di un tipico sistema di ispezione su nastro [8]

1) Sistema di illuminazione:

La qualità delle immagini acquisite gioca un ruolo fondamentale nel semplificare il problema di ispezione. Batchelor [10] ha eseguito uno studio completo di vari schemi di illuminazione per l'ispezione visiva automatizzata (AVI). Esistono quattro schemi di illuminazione maggiormente utilizzati per l'ispezione visiva: anteriore, posteriore, fibra ottica, strutturata.

La retroilluminazione, dove applicabile, elimina gli effetti di ombra e abbagliamento quando utilizzata per ispezionare tessuti. Anche la fibra ottica da buoni risultati in quanto fornisce una illuminazione uniforme, senza problemi di ombre e riflessi, ma ha lo svantaggio di essere più costosa da realizzare. L'uso della luce incidente sul componente da analizzare è di fondamentale importanza anche per altri tipi di produzione, compresa quella dei materiali compositi, tema approfondito nel Capitolo 4 nel parlare della creazione dei diversi dataset.

2) Telecamera:

Viene da sé come la scelta della telecamera sia un fattore di assoluta rilevanza da tenere in considerazione nel progettare un sistema di ispezione intelligente. Ne esistono una gran varietà con differenze relative al tipo di sensore di cui fanno uso, risoluzione, velocità di lettura, precisione e altre caratteristiche che trovano applicazione nella visione artificiale.

La risoluzione di una telecamera è limitata dal numero di pixel nel fotosensore e dal campo visivo dell'oggetto. Quest'ultimo è dipendente dalle caratteristiche dello sfondo e dalla natura dei difetti da rilevare.

Due sono le principali tipologie di telecamere usate per la scansione: telecamere lineari e telecamere matriciali. Le prime adottano un sistema di fotosensori disposti linearmente e la risoluzione nella direzione verticale è funzione della velocità del movimento dell'oggetto e della velocità di scansione alla quale la telecamera sta operando.

Esattamente come uno scanner, acquisiscono una riga di pixel alla volta. A differenza di quest'ultimo però la scansione avviene sempre nello stesso punto, rendendole particolarmente adatte ad acquisire oggetti in movimento continuo, assicurando in ogni punto lo stesso angolo relativo tra telecamera ed illuminatore. Moderne telecamere a scansione lineare di solito forniscono risoluzioni molto elevate e possono ispezionare un'ampia porzione di nastro tessile in un'unica riga di scansione.

In tutte le scansioni in linea, per garantire la sincronizzazione tra la velocità di scansione della telecamera e la velocità di trasporto, è sempre necessario l'utilizzo di un encoder.

Lo svantaggio che si ha nell'utilizzare tali telecamere è che non consentono di generare un'immagine completa in un'unica scansione e necessitano di un hardware esterno per ottenere un'immagine a partire da scansioni multiple.

Le telecamere matriciali, invece, sono in grado di ricostruire un'immagine bidimensionale, non necessitano di un encoder e la risoluzione dell'ispezione in entrambe le direzioni è indipendente dalla velocità dell'oggetto. Il sensore che permette di acquisire l'immagine può essere realizzato con due tecnologie, che prendono il nome di CCD e CMOS.

Le telecamere lineari hanno una risoluzione spaziale molto alta e trovano maggiore impiego nelle produzioni industriali che hanno a che fare con superfici piatte (es. tessile) e in cui sia necessario acquisire immagini ad alta definizione. Le matriciali, invece, non hanno la stessa qualità ma consentono di ottenere una maggiore resa della profondità e dello spazio, e vengono usate in tutte le applicazioni in cui sia richiesta una migliore percezione della distanza e dei particolari.

Di norma il costo di una telecamera lineare è molto elevato rispetto ad una matriciale e spesso gruppi di queste ultime sono utilizzate per fornire soluzioni più economiche al problema di ispezione.

3) Encoder:

La funzione dell'encoder è di fornire il segnale per attivare la telecamera. Nel caso di telecamere a scansione lineare, la risoluzione dell'encoder, ovvero il numero di impulsi per giro, determina la risoluzione dei pixel. Le fotocamere a scansione lineare possono acquisire immagini nitide a qualsiasi velocità adattando la frequenza di scansione della fotocamera alla velocità di trasporto.

4) Frame grabber:

Le informazioni dei pixel provenienti da ciascuna delle fotocamere sono convertite in immagini digitali dal frame grabber. Tutti i sistemi di ispezione di rotoli continui, come quelli utilizzati per i tessuti, devono gestire i diversi ingressi multipli della telecamera. Alcuni lo fanno utilizzando una sorta di multiplexer per video tra la telecamera e il frame grabber. Un metodo alternativo ma molto costoso consiste nell'utilizzare un frame grabber per telecamera [11]. Ciò permetterebbe, in un sistema dotato di più processori, di elaborare parallelamente i dati dei pixel dell'immagine.

5) Host computer:

Le funzioni dell'host computer possono essere riassunte nei seguenti tre punti:

1) Rilevamento dei difetti e classificazione: i dati dell'immagine dai frame grabber vengono scaricati nel computer che, tramite complessi algoritmi, si occupa della loro elaborazione. Segue una fase di classificazione dei difetti rilevati in base alla loro sede di origine e dimensione.
2) Illuminazione e controllo della telecamera: tramite computer è possibile effettuare il caricamento dei parametri di controllo della telecamera, spesso attraverso interfacce utente (GUI). Si può inoltre regolare il livello di illuminazione del nastro trasportatore.
3) Controllo del sistema: tramite computer è possibile eseguire funzioni di controllo di input/output.

3.3 Tecniche tradizionali

I sistemi di visione artificiale per l'individuazione di eventuali difettosità presenti sulle trame dei tessuti fanno uso di un gran numero di tecniche per l'analisi delle texture e la loro classificazione.
L'approccio più promettente e accurato per descrivere una texture è estrarne le caratteristiche uniche. Una panoramica delle principali tecniche tradizionali adottate nei sistemi di visione artificiale viene riportata da Czimmerman et al. [12], basandosi su di una revisione di numerosi articoli pubblicati nelle maggiori riviste scientifiche del settore. Tale divisione, inizialmente proposta da Xie et al. [13] e successivamente Ngan et al. [14], classifica le tecniche per estrarre le caratteristiche dalla trama in quattro approcci, vale a dire: statistico, strutturale, basato su filtri, basato sul modello.

1) Statistico:
Gli approcci statistici si concentrano sull'analisi della distribuzione spaziale dei valori dei pixel in un'immagine acquisita. Fanno parte di questa categoria numerose tecniche tra le quali possono essere citate: statistiche basate su istogrammi, matrice di co-occorrenza, autocorrelazione, local binary pattern (LBP) e altri.
Le proprietà e le statistiche dell'istogramma hanno come punti di forza la semplicità e invarianza a rotazioni e traslazioni. Questi processi includono operazioni statistiche, come intervallo di sensibilità, media, media geometrica, media armonica, deviazione standard, varianza e mediana.
Le matrici di co-occorrenza del livello di grigio (GLCM) rappresentano una delle features maggiormente usate. Si tratta di metodi che misurano la relazione spaziale di pixel in scala di grigi all'interno di matrici di co-occorrenza.
Le funzioni GLCM caratterizzano la trama di un'immagine calcolando quanto spesso coppie di pixel con valori specifici e in una relazione spaziale specificata si verificano in un'immagine, dato un vettore di spostamento, e in seguito estraendo le caratteristiche della trama come energia, entropia, contrasto, omogeneità e correlazione da queste matrici.
I metodi che utilizzano l'autocorrelazione possono essere applicati su trame che contengono modelli ripetitivi, come nel caso del tessile. Questo algoritmo mira a trovare una correlazione tra la trama e la sua traduzione con un vettore di spostamento e ricava i vertici in caso di elevata regolarità.
Il local binary pattern (LBP) è una tecnica per rilevare texture a basso costo computazionale e molto efficiente. Calcola soglie dei pixel in scala di grigi e dei suoi vicini in una finestra scorrevole, che

utilizza il pixel centrale della finestra come soglia. L'LBP è inoltre insensibile ai cambiamenti di illuminazione e alla rotazione dell'immagine, il che lo rende un operatore robusto.
Per citare alcuni esempi applicativi, Zhang et al. [15] hanno combinato i metodi GLCM e LBP per estrarre le features dell'immagine per addestrare una rete neurale basata sulla backpropagation, raggiungendo un tasso di rilevamento del 97,6% su 90 campioni dal database TILDA [16]. In un altro studio [17], hanno proposto un metodo LBP modificato per addestrare un classificatore SVM, rilevando i difetti con un'accuratezza del 93% su 148.905 campioni.

2) Strutturale

Gli Approcci Strutturali si concentrano principalmente sulla posizione spaziale degli elementi della texture, che possono essere estratti e descritti come primitive di trama. Si tratta per lo più di semplici regioni in scala di grigi, segmenti di linea o singoli pixel. Questi elementi sono sempre utilizzati in una combinazione con regole di posizionamento derivate dalle relazioni geometriche o dalle statistiche spaziali delle primitive. Questo approccio trova la sua più alta espressione se applicato su trame regolari.
Sono stati sviluppati diversi metodi per gli approcci strutturali, come ad esempio quello proposto da Chen e Jain [18] che descrive un modello con una struttura a scheletro della trama o quello proposto da Bennamoun e Bodnarova [19] che descrive un approccio chiamato texture blob detection: entrambi sono realizzati per il rilevamento dei difetti sulle immagini dei tessuti.

3) Basato su filtri:

Tramite questo approccio le immagini possono essere descritte dalle caratteristiche rilevate, come bordi, trame e regioni. Il filtraggio di queste features ha rappresentato uno dei primi tentativi di elaborazione delle immagini, soprattutto per l'estrazione di dettagli dei bordi, per il cui scopo è possibile utilizzare diversi filtri e algoritmi nel dominio spaziale, come i filtri di Sobel, Robert, Canny, Deriche, Laws e Laplaciani.
Nella maggior parte dei casi, operare nel dominio spaziale comporta rumore e complicazioni per trovare un Kernel. Pertanto, una tecnica largamente usata consiste nel trasformare le immagini nel dominio della frequenza con la trasformata di Fourier (FT), filtrare facilmente il rumore ed elaborare l'immagine, per poi ritrasformarla nel dominio spaziale.
Un problema della trasformazione di Fourier è che dipende dall'intera immagine a causa dei suoi coefficienti. Questa proprietà la rende incapace di localizzare i difetti nel dominio spaziale. La

soluzione più comune per questo problema consiste nell'applicare una Trasformata di Fourier a finestra per la dipendenza spaziale e, se la funzione finestra è gaussiana, risulta nella trasformata di Gabor.

Similmente, anche la trasformata Wavelet è utilizzata come rilevatore di difetti. Basa il suo funzionamento su piccole onde di frequenza variabile e durata limitata chiamate wavelet, che forniscono informazioni locali da direzioni orizzontale, verticale e diagonale dell'immagine.

Neubauer [20] ha introdotto un metodo con tre filtri FIR 5×5 come statistiche del primo ordine ed ha eseguito la segmentazione dei difetti con sensitività del 98,3% e specificità del 90,6%.

Recentemente, Shan Gai [21] ha eseguito la trasformata wavelet su 10.000 immagini di banconote e ha ottenuto una percentuale di successo nel rilevamento dei difetti del 97% e uno 0,35% di falsi allarmi. Kumar e Pang [22] hanno utilizzato solo funzioni Gabor reali per il rilevamento dei difetti nei tessuti, mentre Yang et al. [23] hanno sviluppato un estrattore di features adattivo basato su wavelet con un rilevatore basato sulla distanza euclidea per immagini di tessuti, che ha raggiunto il 97,5% con un database di difetti (480 campioni privi di difetti e 480 difettosi) e il 93,3% senza un database di difetti (780 campioni privi di difetti e 180 campioni difettosi).

4) Basato sul modello:

I metodi basati su modelli sono classificati in tre gruppi: modelli frattali, modelli di campi casuali e modelli autoregressivi.

I primi giocano un ruolo significativo nella descrizione delle superfici naturali con texture irregolare. Gli approcci dei campi casuali combinano informazioni sia statistiche che strutturali di entità dipendenti dal contesto, come i pixel che dipendono dai pixel vicini, e possono essere utilizzati per la segmentazione di texture o per problemi di classificazione.

I modelli autoregressivi hanno come concetto principale quello di caratterizzare le features della trama in base alle dipendenze lineari dei pixel.

Bu et al. [24] hanno eseguito il rilevamento dei difetti sulla base di quattro caratteristiche frattali principali, facendo uso di sette dataset di 14.378 campioni privi di difetti e 3222 campioni di difetti, con una percentuale di successo del 98,3%. In [25], gli autori hanno utilizzato un modello autoregressivo monodimensionale e una telecamera CCD per un'ispezione di rotoli continui di materiale in tempo reale, mentre in [26], gli autori hanno utilizzato un modello quantitativo che caratterizza l'impatto dell'illuminazione con un semplice classificatore e hanno ottenuto una precisione del 94% in 1865 campioni.

3.4 Tecniche di deep learning

Il deep learning rappresenta uno dei campi in più rapida crescita nelle scienze informatiche, grazie alla sua capacità di risolvere problemi di particolare complessità [27]. Tale caratteristica lo rende una delle tecniche maggiormente utilizzate nel realizzare sistemi di visione per l'ispezione visiva di superfici di varia natura. Gli approcci menzionati precedentemente sono considerati soluzioni tradizionali, in cui l'obiettivo degli algoritmi sta nel cercare caratteristiche esplicite, che possono essere difficili da ottenere in determinati contesti. Al contrario, gli algoritmi di deep learning trasformano i dati in complesse e astratte rappresentazioni che permettono al sistema di apprendere le features di interesse.

Nel campo dell'individuazione di difetti superficiali, le tecniche di deep learning basate su reti neurali che hanno riscosso maggior successo possono essere suddivise nelle seguenti categorie: convoluzionali, autoencoder, deep residual, fully convolutional, ricorrenti.

Una descrizione dei precedenti metodi è riportata da Yang et al. [28], anche in questo caso basandosi su di una revisione di numerosi articoli che trattano l'argomento e nella tabella 1 ne sono sintetizzati vantaggi e svantaggi.

Metodi	Punti di forza	Debolezze	Campi di applicabilità
CNN	Possiedono una forte capacità di apprendimento per dati in ingresso a più dimensioni; sono in grado di apprendere features astratte, semplici e di alto livello da pochi dati preprocessati o originali.	La complessità computazionale è notevolmente incrementata con l'aumentare della profondità della rete.	Qualunque tipo di materiale.
Autoencoder neural network	Sono in grado di rappresentare ad un buon livello l'informazione di oggetti; possono estrarre la regione in primo piano rispetto ad uno sfondo complesso; buona robustezza ai rumori ambientali.	La dimensione dei dati in ingresso e uscita dell'autoencoder deve essere coerente.	Qualunque tipo di materiale.
Depth residual neural network	In genere presentano migliori prestazioni di classificazione dovute ad una minore loss e capacità di non causare overfit.	Per sfruttare il vantaggio dato dalla sua struttura sono necessarie reti molto profonde.	Qualunque tipo di materiale.
Fully convolutional neural network	Può estrarre feature da immagini di qualsiasi dimensione e ottenere matrici di conoscenza semantica di alto livello, che hanno un buon riscontro nella object detection a livello semantico.	Necessita di una matrice di trasformazione di feature combinata con le feature sottostanti, e hanno una lenta velocità di convergenza	Qualunque tipo di materiale.
Recurrent neural network	Quando ci sono pochi dati, consente di apprendere le caratteristiche essenziali e ridurre la loss nel processo di pooling.	Con l'aumentare del numero di iterazioni nella fase di addestramento della rete, può andare incontro ad overfitting più facilmente.	Qualunque tipo di materiale.

Tabella 1: Metodi di deep learning per l'individuazione di difetti [28]

1) Convolutional neural network:

Le CNN rappresentano una tipologia di rete neurale caratterizzata da tre strati principali: strati convoluzionali, strati di pooling, strati densi (fully connected). Lo strato convoluzionale applica una serie di filtri nelle regioni locali dell'input, costruendo la mappa delle feature dell'immagine passata in ingresso. Lo strato di pooling esegue un sottocampionamento lungo le dimensioni spaziali dell'input ed è normalmente applicato dopo lo strato convoluzionale per ridurre la dimensione delle feature e evitare problemi di over-fitting. Gli strati densi costituiscono gli ultimi layer di una CNN ed eseguono il task di classificazione prendendo la decisione finale. Una CNN profonda è normalmente caratterizzata da un'alternanza di strati convoluzionali e di pooling, seguiti dai densi. Una descrizione più approfondita verrà fatta nel Capitolo 4 in un paragrafo dedicato, in cui saranno presentate le principali reti utilizzate per la soluzione proposta.

2) Autoencoder neural network:

Una rete autoencoder è costituita da due fasi principali: la fase di codifica e di decodifica. Nella fase di codifica, il segnale di ingresso è convertito in un segnale per l'estrazione delle feature; nella fase di decodifica, l'informazione ottenuta da queste ultime è convertita in un segnale per la ricostruzione e successivamente l'errore di ricostruzione è minimizzato aggiustando i pesi e il bias per ottenere il risultato finale di identificazione dei difetti. Una differenza sostanziale tra le reti autoencoder e gli altri algoritmi di machine learning è che l'obiettivo delle prime sta nell'apprendimento delle features, non nel classificare.

3) Deep residual neural network:

La deep residual va ad aggiungere un modulo residuale alla base della rete neurale convoluzionale. Si caratterizza per una facile ottimizzazione e può migliorare l'accuratezza aumentando la profondità della rete. A differenza delle CNN o Generative Adversarial Network (GAN) per le quali all'aumentare della profondità si ha un aumento della capacità di estrarre features, con il rischio di non far convergere la funzione di attivazione, lo scopo delle deep residual è di ottimizzare l'incremento del numero di strati della rete con i residui e allo stesso tempo aumentarne la struttura; basano il proprio funzionamento sul concetto di "skip connection", che consente di trasferire l'informazione dell'output di uno strato ad un altro non adiacente più in profondità nella rete, permettendo di aggiungere strati senza degradare l'informazione. Questo concetto sarà approfondito nel Capitolo 4 quando si introdurrà la ResNet, primo modello a farne uso.

4) Fully convolutional neural network:

Una fully convolutional neural network (FCNN) è una rete neurale profonda priva di strati densi finali e che fa uso di convoluzioni con Kernel 1x1 per implementare gli strati fully connected, riuscendo in questo modo a gestire input di dimensione diversa. Ciò le differenzia dalle CNN classiche, che sono invece dotate di strati fully connected che bloccano le dimensioni dell'input, costringendo ad un ridimensionamento nel caso in cui differiscano da quelle consentite. Inoltre, l'assenza di strati densi consente di avere una struttura con un minor numero di parametri e aumentare di conseguenza la velocità computazionale, riducendo la latenza.

5) Recurrent neural network:

Le reti neurali ricorrenti sono reti neurali che hanno connessioni cicliche tra i neuroni anziché in una sola direzione come le reti feed forward, quali sono le CNN. In altri termini, nelle reti feed forward l'input di uno strato dipende solo dall'output degli strati precedenti. Al contrario, una rete ricorrente mette in atto un meccanismo di memoria, dato che l'input di uno strato è costituito non solo dall'output dello strato che lo precede spazialmente, ma anche da sequenze di input precedenti temporalmente e già esaminate.

Un'altra caratteristica distintiva delle reti ricorrenti è che condividono i parametri su ogni livello della rete, anche in questo caso differenziandosi dalle feed forward che hanno pesi diversi per ciascun nodo. I pesi sono comunque ancora regolati tramite backpropagation e discesa del gradiente. Queste reti sfruttano l'algoritmo di backpropagation nel tempo (BPTT), che si differenzia dal metodo tradizionale in quanto somma gli errori in ogni fase temporale, operazione non necessaria per reti feed forward in quanto non c'è condivisione dei parametri su ogni livello. Questo processo genera però due problemi, noti come gradiente "esplosivo" e "evanescente". Il primo si verifica quando il gradiente è troppo grande, generando un modello instabile; il secondo quando è troppo piccolo, con un gradiente che continua a ridursi aggiornando i parametri fino a valori insignificanti.

Un'altra distinzione che può essere fatta riguarda il tipo di problema che si tenta di risolvere, ovvero di classificazione, segmentazione o detection. Sono infatti presenti nella letteratura numerosi esempi di reti neurali adatte per ognuno dei precedenti scopi.

Brackenbury et al. [29] hanno confrontato tre diverse strategie di classificazione di deep convolutional neural network (DCNN) per rilevare difetti strutturali nei ponti ad arco in muratura. Il loro studio mostra l'importanza della scelta del metodo più adatto in base al tipo di dataset.

Diversi studi hanno riguardato l'individuazione di difetti in tessuti [30-32]. Nello studio [30] hanno provato a individuare difetti adottando una faster RCNN combinata con una ResNet 101 per la fase di estrazione delle features, raggiungendo una precisione media di oltre il 94%.

Zhu et al. [31] hanno utilizzato una densenet modificata per fare detection di 11 classi di difetti nei tessuti, ottenendo una capacità media nell'identificarli dell'83%. Liu et al. [32] propongono un approccio innovativo nel tentativo di risolvere la problematica di dataset limitati. A partire da un esiguo numero di immagini acquisite in ambiente industriale e da immagini di difetti generate tramite GAN, hanno utilizzato una DeepLAbV3 pre-addestrata per la segmentazione, ottenendo una precisione media del 94,8%, recall media del 97,6% e F-measure del 96,2%. Altri approcci hanno riguardato l'industria del legno [34][35] o del metallo [34][36].

Urbonas et al. [34] affrontano un problema di identificazione di difetti nel legno tramite una faster RCNN testando varie reti pre-addestrate per l'estrazione delle features, con un'accuratezza media dell'80,6% per quattro classi di difetto e 96,1% per una classe di difetto.

Gao et al. [35] hanno sviluppato un algoritmo composto da tre VGG16 pre-addestrate che lavorano in maniera indipendente per l'estrazione delle features e in seguito operano una fusione dei risultati per il riconoscimento. Testando due differenti dataset hanno raggiunto un'ottima accuratezza rispettivamente del 99% per il dataset NEU (metallo) e dell'88,28% per il legno. Lo stesso dataset NEU è stato testato precedentemente da Chen et al. [36] combinando una resnet con due WRN e raggiungendo un'accuratezza del 99.89%.

Cha et al. [37] hanno utilizzato un metodo di ispezione visiva basato su Faster R-CNN per rilevare e classificare cinque tipi di difetti con precisioni medie del 90,6%, 83,4%, 82,1%, 98,1% e 84,7%. In particolare, il loro metodo ha svolto il compito in modo significativamente più rapido rispetto a un metodo tradizionale basato sulla CNN, caratteristica necessaria per una eventuale implementazione in tempo reale. Wang et al. [38] hanno sviluppato un algoritmo di faster R-CNN per risolvere il problema della velocità delle CNN e per individuare piccoli difetti in prodotti geometricamente complessi dove hanno raggiunto il 72% di rilevamento e l'81% di precisione nella classificazione. Liu et al. [39] ha introdotto un metodo per il rilevamento dei difetti basato sulla segmentazione semantica. Per questo scopo, hanno fatto uso delle sopra citate Fully Connected Networks (FCN), raggiungendo una precisione del 99,6% sul dataset DAGM 2007 tedesco.

Recentemente, Yang et al. [40] hanno sviluppato un sistema basato su DCNN per rilevare e classificare i difetti che possono verificarsi durante la saldatura laser nella produzione di batterie. Oltre a ciò, hanno proposto un modello di VGG per migliorare l'efficienza della classificazione dei difetti. I loro test su 8000 campioni con una precisione del 99,87% hanno dimostrato che il modello VGG pre-addestrato ha dimensioni ridotte, minore tasso di falsi positivi e tempi di addestramento e di previsione più brevi, rendendo il loro modello particolarmente adatto per il controllo qualitativo in ambiente industriale.

3.5 Visione artificiale applicata ai compositi

Dalla ricerca in letteratura sono emersi diversi studi che trattassero il rilevamento di difetti in materiali compositi, tuttavia non prendendo in considerazione difetti superficiali e di natura estetica, ma strutturali e presenti all'interno del materiale. Come conseguenza, non è stato possibile reperire un dataset di interesse da cui partire, introducendo un ulteriore elemento di criticità da tenere in considerazione. Per completezza, nel seguito verranno brevemente riportati alcuni risultati ottenuti con questi approcci esclusivamente per i materiali compositi.

Una delle tecniche maggiormente utilizzate dai ricercatori, consiste nell'addestrare i propri modelli con immagini termografiche [41 - 46]. Bang et al. [41] propongono un framework per identificare difetti strutturali in materiali compositi a partire da un dataset di immagini presenti in letteratura, impiegando una faster RCNN unita ad una Inception V2 e una Inception Resnet V2 per l'estrazione di features, raggiungendo una precisione media del 75%. Numan et al. [42] propongono un algoritmo basato su CNN combinato con una Deep Feed Forward (DFF NN) per rilevare automaticamente difetti strutturali e stimarne anche la profondità. L'utilizzo di una rete pre-addestrata con dataset quali CIFAR-10 e ImageNet, ed in seguito perfezionata tramite fine tuning con un dataset relativamente piccolo, ha consentito di raggiungere una accuratezza di circa l'88%. Un altro risultato importante lo ottengono Marani et al. [43] presentando un nuovo metodo che approssima i decadimenti termici sulla superficie del laminato, indotti da un breve impulso di calore, per mezzo di un modello esponenziale a tre incognite, stimato ai minimi quadrati. Test sperimentali effettuati su un polimero rinforzato con fibra di carbonio (CFRP) hanno raggiunto un'accuratezza standard e bilanciata rispettivamente del 99,47% e 86,9%, mentre precisione e recall sono rispettivamente dell'89,87% e 73,67%. Lo stesso autore [44] recentemente presenta un'altra procedura per l'analisi non distruttiva dei laminati compositi, combinando analisi termografiche a infrarossi e reti neurali profonde.

In [45] gli autori insistono sulla tematica dei test non distruttivi per rilevare regioni difettose nelle strutture di polimeri rinforzati con fibra di carbonio (CFRP), estendendo l'utilizzo dei metodi dei minimi quadrati. I risultati sperimentali mostrano che, con questa strategia, le regioni con difetti nelle immagini termografiche risultano più nitide, mentre i valori del rapporto segnale-rumore (SNR) aumentano in modo significativo. Nel recente studio effettuato da Liu et al. [46] viene presentato un nuovo metodo denominato "Generative Kernel Principal Component Thermography" (GKPCT) per la valutazione dei difetti interni dei materiali compositi. Quest'ultimo sfrutta una generative adversarial network per aumentare i termogrammi per la costruzione del modello.
Successivamente, il metodo KPCT viene usato per mappare le features dei dati termografici tramite l'analisi delle componenti principali.
In [47] Margraf et al. presentano uno studio per la scelta semi-automatica dei migliori parametri da utilizzare nel segmentare difetti filamentosi, tramite un approccio basato sulla programmazione genetica cartesiana (CGP). Infine, Gong et al. [48] descrivono un modello di deep learning che, sfruttando la tecnica del transfer learning, estrae con precisione le features che indicano la presenza di difetti in immagini ai raggi X di materiali compositi per uso aereonautico (ACM). Con tale metodo sono riusciti ad ottenere un F1 score del 96%.
Complessivamente, da questa ricerca è comunque possibile trarre dei validi spunti su come procedere per il lavoro richiesto. Considerando la novità rappresentata dal problema da risolvere e il loro sempre crescente utilizzo per casi simili, si è deciso di adottare tecniche di deep learning, nello specifico di reti neurali convoluzionali. I dettagli del metodo proposto e i risultati ottenuti su dei dataset sperimentali di immagini con e senza difetti sono riportati nei seguenti Capitoli 4 e 5 della trattazione.

4 Metodo proposto

Nei capitoli precedenti è stato più volte riportato come il problema del riconoscimento automatizzato di difetti superficiali in materiali compositi non sia stato ancora affrontato in ambito di ricerca. La decisione riguardo la metodologia da adottare per affrontare il problema è dipesa da diversi fattori, tra i quali rientrano:

1) Richieste iniziali avanzate dall'azienda ospitante.
2) Difficoltà affrontate nella fase di approccio al problema.
3) Scelta della tecnica migliore in base al contesto.
4) Scelta degli strumenti software da utilizzare.

Il primo punto fa riferimento agli obiettivi inizialmente posti e da portare a termine con il presente lavoro. Questi ultimi riguardano l'individuazione di difetti estetici presenti in materiali compositi al termine della fase di produzione. Il secondo punto fa riferimento alle principali difficoltà riscontrate nel periodo iniziale di tirocinio, durante il quale vi era la necessità di capire come si sarebbe tentato di risolvere il problema. Tra le considerazioni che hanno influenzato il successivo metodo proposto, rientrano sicuramente:

1) L'assenza di dataset costituiti da immagini di superfici di materiali compositi, con e senza difetti. Ciò rende necessario la realizzazione di uno o più dataset sperimentali a partire da immagini acquisite da componenti prodotti in azienda. Data l'importanza dell'argomento, in seguito saranno dedicati diversi paragrafi nei quali verranno descritte le principali strategie adottate per l'ottenimento di immagini di qualità con le risorse a disposizione.

2) La natura dei componenti da analizzare: la particolarità del compito richiesto è data dalla sostanziale differenza riscontrabile tra le superfici in esame e quelle normalmente analizzate per problematiche simili affrontate in letteratura. La maggior parte dei sistemi automatizzati per il controllo della qualità, infatti, è utilizzato per ispezionare superfici piane o dalle geometrie semplici. Nel presente contesto, i componenti prodotti dall'azienda sono caratterizzati dall'avere forme e dimensioni molto diverse, con pezzi che vanno da pochi centimetri a qualche metro di lunghezza, in base alla tipologia di componente da produrre e al mercato di destinazione. Inoltre, un altro

elemento di diversità è rappresentato dalla texture delle fibre di carbonio tra loro intrecciate, a seconda della tecnica di tessitura impiegata. Infine, vale la pena menzionare l'importanza dell'illuminazione cui sono soggette le superfici in esame: in base alle condizioni di luce e all'angolo di incidenza tra illuminatore e pezzo da esaminare, la natura dei materiali compositi fa sì che si generino numerosi riflessi e la resa visiva degli eventuali difetti presenti sia differente al variare delle condizioni.

Dopo aver individuato le problematiche iniziali ed aver effettuato la ricerca in letteratura su tematiche analoghe, il passo successivo ha riguardato la scelta della tecnica più adatta per eseguire il task richiesto. Considerata l'unicità della texture delle superfici, l'ampia variabilità dei componenti, la differente resa delle immagini ottenibili al variare delle condizioni di luce e la tipologia di difetti da rilevare, si è deciso di adottare tecniche di deep learning, nello specifico di reti neurali convoluzionali, al posto dei metodi tradizionali di visione artificiale. La capacità delle CNN di poter estrarre quasi tutte le features di interesse dall'immagine le rende particolarmente adatte allo scopo e, non a caso, sono tra i metodi maggiormente utilizzati nel realizzare i moderni sistemi di ispezione. Come sarà approfondito in seguito, sono stati testati numerosi modelli di reti presenti in letteratura e largamente utilizzati per problemi di classificazione a partire da immagini, pre-addestrati su dataset contenenti milioni di immagini ed in seguito perfezionati per essere adattati al problema in esame.
Il quarto punto riguarda le tecnologie adottate per la scrittura del codice e l'esecuzione dei test per la validazione dei modelli proposti, per i quali verrà dedicato un paragrafo al termine del presente capitolo.
Terminata la fase iniziale, fondamentale per impostare il lavoro da svolgere nei successivi mesi, si è meglio delineato il percorso da seguire per il raggiungimento dello scopo finale.
Per una maggiore chiarezza, il metodo proposto, seguendo la procedura largamente utilizzata per tecniche di deep learning, può essere sintetizzato nei seguenti tre punti:

1) Costruzione del dataset.
2) Realizzazione di più modelli di deep learning.
3) Addestramento dei modelli, confronto e valutazione dei risultati.

I precedenti punti verranno affrontati in dettaglio nel Capitolo 4 e 5 della trattazione.

4.1 Dataset

La prima operazione da effettuare è stata la creazione del dataset da cui partire per poter addestrare le reti neurali scelte per il problema di classificazione. Come si avrà modo di comprendere nei paragrafi successivi, è stato possibile realizzare ben tre differenti dataset nel corso del tirocinio. Questa scelta è dipesa principalmente da due motivazioni, legate alla strategia adottata e alle risorse di cui ho potuto fare uso nel periodo di tirocinio. La prima motivazione è giustificabile considerando la novità rappresentata dal lavoro e la necessità di procedere per step: non potendo conoscere a priori il comportamento dei modelli sviluppati, si è preferito cominciare dal problema più semplice, vale a dire creare un dataset binario e classificare le immagini in prive di difetti e con difetti. Successivamente, avendo constatato la fattibilità delle soluzioni proposte, unitamente alla maggiore disponibilità di componenti da cui ottenere le immagini, si è provveduto ad estendere il numero di classi a tre per i successivi dataset. La seconda motivazione è invece di carattere pratico, avendo a che fare con la tecnologia utilizzata per ottenere le immagini. I tre dataset creati, infatti, sono popolati da immagini ottenute tramite smartphone, non disponendo nei primi mesi di tirocinio di strumenti più adeguati per lo scopo. Successivamente, grazie alla collaborazione con l'azienda Vision Device di Torrevecchia Teatina (CH) [49], specializzata nella realizzazione di sistemi di visione artificiale e in possesso della tecnologia necessaria per operare in scenari industriali, si è valutata la possibilità di realizzare un nuovo dataset con immagini più standardizzate e di qualità, argomento riservato a possibili sviluppi futuri e approfondito nel Capitolo 6.

Nei prossimi paragrafi verranno descritte nel dettaglio le caratteristiche di ognuno dei dataset menzionati, unitamente agli accorgimenti adottati per ottenere immagini valide a seconda della tecnologia che avevo a disposizione al momento.

4.1.1 Dataset binario

Come anticipato nel paragrafo precedente, si è deciso di approcciare il tema in esame con cautela. Infatti, sebbene i modelli sviluppati fossero basati su alcune delle architetture di rete neurale convoluzionale più utilizzate per il problema di classificazione di difetti superficiali e dai risultati più promettenti, non vi era la sicurezza che potessero comportarsi allo stesso modo se impiegate su superfici particolari quali sono quelle oggetto di studio. Dunque, come primo tentativo si è deciso di realizzare un dataset binario, da utilizzare per addestrare i modelli implementati e ottenere dei

primi risultati utili per capire le potenzialità del sistema pensato. Per questo primo dataset sono state realizzate due classi, denominate “negative” e “positive” ad indicare immagini rispettivamente prive di difetti e con difetti. In questa fase, non disponendo ancora di un adeguato numero di componenti da cui ottenere le immagini e non essendo stati in grado di poter osservare tutte le tipologie di difetti presentate nel Capitolo 2, si è preferito considerare difetto la sola discontinuità di trama, che rappresenta comunque una delle tipologie più diffuse e importanti da rilevare, data l’impossibilità di porvi rimedio se non tramite verniciatura. Si è invece deciso di non considerare difetto, facendola rientrare nella classe “negative”, la presenza di porosità isolate, presente in gran numero ma caratterizzata dall’essere facilmente risolvibile.

Avendo la possibilità di gestire la costruzione del dataset, si è inoltre deciso di inserire lo stesso numero di immagini per classe, in modo da disporre di un dataset bilanciato e ottenere dalle reti neurali una migliore generalizzazione. In questa prima versione ogni classe è composta da 200 immagini, per un totale di 400. Come verrà descritto nel Capitolo 5, per ogni dataset realizzato è stata prevista una ulteriore versione contenente il triplo delle immagini, ottenute tramite tecniche di data augmentation, raggiungendo per il dataset binario 600 immagini per classe e 1200 totali.

Un altro aspetto di notevole importanza riguarda la qualità delle immagini inserite. Infatti, in questa fase non si disponeva di strumenti quali telecamere lineari e/o matriciali o sistemi di illuminazione fissi, pertanto per acquisire le varie foto si è utilizzato uno smartphone. Come è facilmente intuibile, questo metodo non consente di ottenere immagini standardizzate, in cui l’angolo relativo tra telecamera, illuminatore e componente rimane costante. Inoltre, come spesso ricordato, le superfici dalle quali ottenere le foto talvolta presentavano curvature e zone poco visibili, comunque sedi di difetti di varia natura e dunque da prendere in considerazione. Di conseguenza, si è deciso di non utilizzare un’unica tecnica di acquisizione, ma di scattare foto da varie angolazioni, distanze e condizioni di luce. In questo modo, non si sarebbe realizzato un dataset con immagini paragonabili a quelle ottenibili con un sistema industriale, ma ciò avrebbe consentito di valutare le potenzialità delle CNN applicate ad un contesto nuovo e a partire da immagini dalla qualità migliorabile.

Per ottenere le immagini finali presenti nel dataset, sono stati previsti i seguenti passaggi:

1) Acquisizione delle immagini tramite smartphone (3968x2976).
2) Zoom per ritagliare porzioni significative dell’immagine originale.
3) Resizing (224x224).
4) Labelling manuale.

A partire dalle foto originali ottenute tramite smartphone con una risoluzione pari a 3968x2976, si è provveduto in una fase successiva a ritagliare delle porzioni delle stesse che contenessero solamente superfici di materiali compositi, includendo anche aree che presentassero curvature non troppo accentuate. In seguito, ogni immagine così ottenuta è andata incontro ad un resizing, portandola ad avere una dimensione di 224x224 pixel, formato ampiamente utilizzato come ingresso per il primo strato delle reti neurali. Infine, si è provveduto a etichettare manualmente le singole immagini, adottando la dicitura "negative_#" e "positive_#" rispettivamente per le immagini che sarebbero state inserite nella classe "negative" e "positive".

Nelle figure sottostanti sono riportati alcuni esempi di immagini utilizzate all'interno del dataset, nelle quali si può notare come la distanza dalla superficie e le condizioni di luce siano variabili. Inoltre, sono visibili anche delle segnature di pennarello, utilizzato dal personale addetto al controllo della qualità per segnalare la presenza di eventuali difetti non recuperabili, che costituiscono un ulteriore elemento di disturbo per il classificatore.

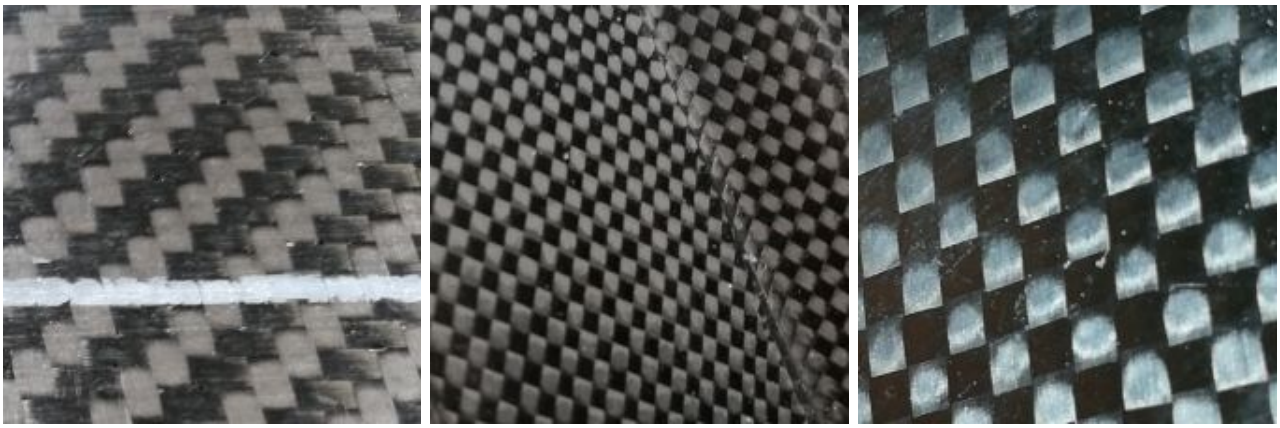

Figura 11: Esempio immagini della classe "negative" - dataset binario

Figura 12: Esempio immagini della classe "positive" - dataset binario

4.1.2 Dataset a tre classi – prima versione

Dopo aver constatato la validità di alcuni dei modelli proposti addestrati con le immagini presenti nel dataset binario, si è tentato di aumentare la complessità del problema di classificazione rendendolo multi-classe. L'idea iniziale consisteva nel creare una classe per ogni tipologia di difetto, tuttavia la grande disparità nella loro insorgenza avrebbe causato la realizzazione di un dataset fortemente sbilanciato. Si è dunque optato per creare un dataset a tre classi, in cui la classe "positive", presente nella versione binaria e contenente i difetti, è stata divisa in due differenti, a rappresentarne le due grandi categorie, vale a dire recuperabili e non recuperabili. Inoltre, la classe "negative", precedentemente popolata da immagini in cui si potevano riscontrare piccole porosità, in questo caso è costituita da sole immagini prive di qualsiasi difetto. In questa nuova versione, le tre classi prendono il nome di "negative", "non_recoverable_defects" e "recoverable_defects", ad indicare rispettivamente immagini prive di qualsiasi difetto, con difetti non recuperabili e con difetti recuperabili. Inoltre, nella classe dei difetti non recuperabili, in aggiunta alla discontinuità di trama, sono state inserite anche immagini di grandi porosità, mentre nella classe dei difetti recuperabili, oltre alle piccole porosità, sono state incluse le infiltrazioni di corpi estranei quali alluminio e polietilene. Una ulteriore modifica rispetto al dataset binario è riscontrabile nel numero di immagini per classe portate a 250, per un totale di 750 per il dataset originale, mentre con la data augmentation si raggiunge un numero di 750 immagini per classe, con 2250 totali. Non cambiano invece la modalità di acquisizione delle foto e i passaggi successivi per ottenere le immagini finali, eccetto per la dicitura usata per l'etichettatura manuale, indicando con "negative_#", "nrd_#" e "rd_#" rispettivamente le immagini inserite nella classe "negative", "non_recoverable_defects" e "recoverable_defects". Le figure sottostanti riportano alcuni esempi per le tre classi.

Figura 13: Esempio immagini della classe "negative" - dataset a tre classi - prima versione

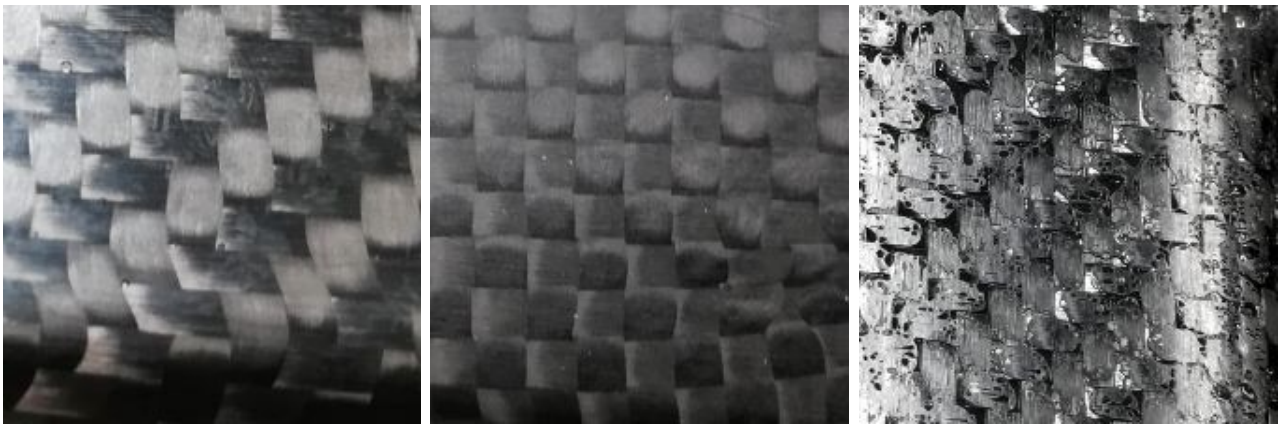

Figura 14: Esempio immagini della classe “non_recoverable_defects” - dataset a tre classi - prima versione

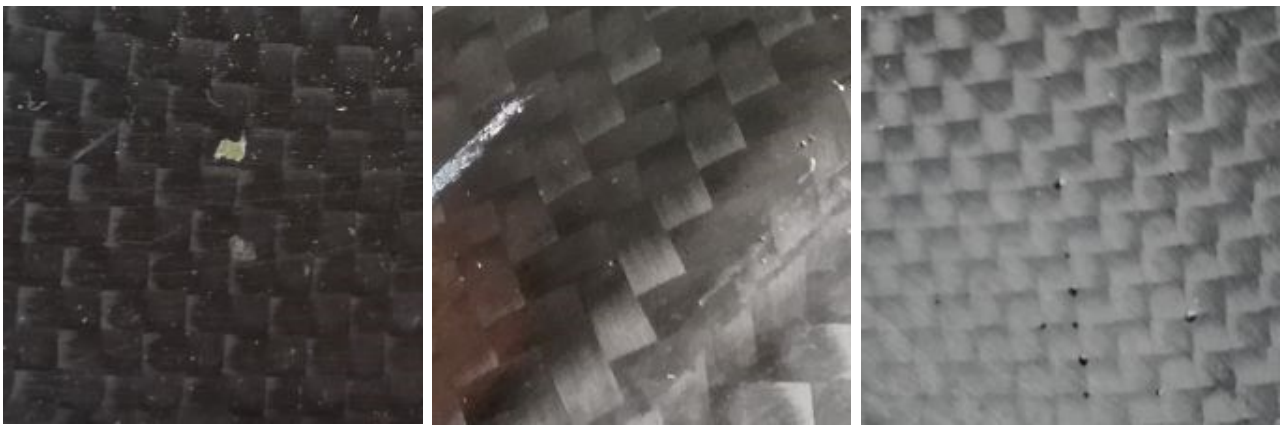

Figura 15: Esempio immagini della classe “recoverable_defects” - dataset a tre classi - prima versione

4.1.3 Dataset a tre classi – seconda versione

La realizzazione di questo nuovo dataset è dipesa dalla disponibilità di nuovi componenti messi a disposizione dall’azienda. In questo modo è stato possibile ottenere numerose foto aggiuntive, comprese alcune relative a nuove tipologie di difetti non presenti nei componenti forniti precedentemente, quali incongruenze visive nel colore della trama e accumulo di resina. Entrambe le categorie sono state incluse tra i difetti non recuperabili, mentre il numero di classi e la dicitura usata nell’etichettatura è rimasta invariata, così come la modalità di acquisizione delle immagini. L’unica differenza consiste nel numero di immagini per classe passate a 500 nella versione originale, per un totale di 1500, e a 1500 nella versione estesa, per un totale di 4500.

Anche in questo caso si riportano degli esempi, tra i quali sono evidenti le due nuove tipologie di difetti per la classe “non_recoverable_defects”.

Figura 16: Esempio immagini della classe “negative” - dataset a tre classi - seconda versione

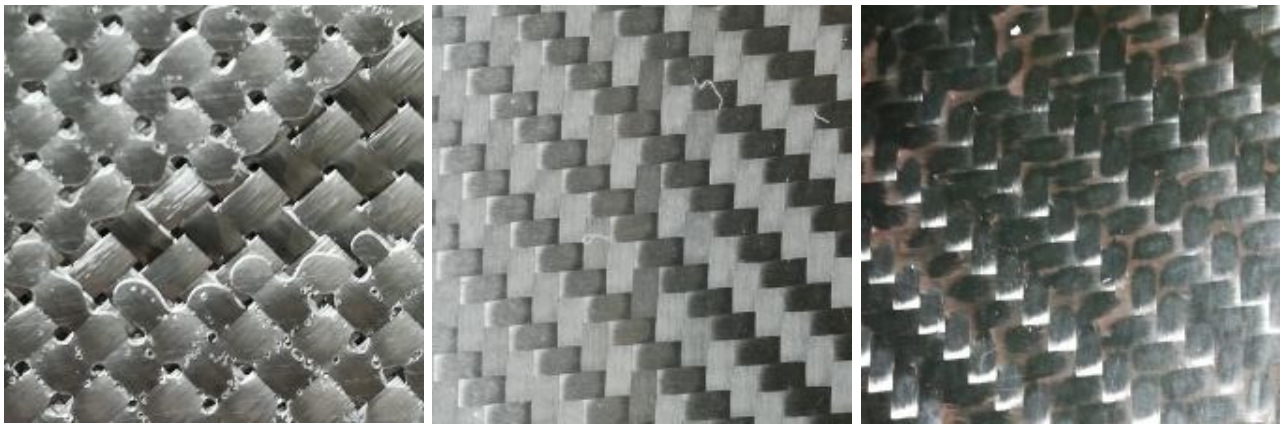

Figura 17: Esempio immagini della classe “non_recoverable_defects” - dataset a tre classi - seconda versione

Figura 18: Esempio immagini della classe “recoverable_defects” - dataset a tre classi - seconda versione

4.2 Classificazione tramite reti neurali convoluzionali

La fase successiva alla costruzione del dataset ha riguardato la scelta del metodo più idoneo per affrontare il problema. Dalla ricerca in letteratura è emerso come le metodologie più utilizzate per l'identificazione di difetti fossero le tecniche tradizionali di computer vision e le più recenti di deep learning. Come più volte riportato nei precedenti capitoli, si è scelto di adottare queste ultime, trasformando il problema di identificazione in un task di classificazione di difetti tramite CNN.

Una rete neurale convoluzionale (CNN) è un tipo di rete neurale originariamente introdotto da Yann LeCun et al. [50] per adottare un approccio basato su reti neurali [51] per il riconoscimento delle cifre scritte a mano. In Figura 19 è rappresentata la tipica architettura di una CNN. L'idea alla base di una CNN è di utilizzare una ripetuta combinazione di strati convoluzionali e strati di pooling [50]. L'output di ogni strato convoluzionale è una operazione di convoluzione con un insieme di finestre mobili di pesi chiamate Kernel o filtri. Le reti multistrato possono essere considerate come approssimatori di funzioni universali per via dell'applicazione di funzioni di attivazione non lineari all'uscita del neurone [52]. Tra le principali funzioni di attivazione possono essere citate la sigmoide, la tangente iperbolica e la rectified linear unit (ReLU). Durante la fase di addestramento, ogni Kernel apprende specifiche feature locali presenti nell'immagine. Per ridurre la dimensione dell'uscita degli strati convoluzionali, si fa uso degli strati di pooling. In genere, l'immagine di output dallo strato convoluzionale viene suddivisa in regioni non sovrapposte e a ciascuna viene applicata una funzione di riduzione. Tra i principali metodi impiegati trovano spazio il Max pooling e l'Average pooling, tramite i quali si ottiene la riduzione andando a prendere rispettivamente il valore più alto e il valore medio dei pesi di ogni regione dello strato precedente. Al termine della serie costituita da strati convoluzionali e di pooling, sono aggiunti diversi strati chiamati fully connected o densi. In altri termini, la CNN sta imparando ad estrarre mappe di feature di alto livello dall'immagine ed eseguire la classificazione finale su quelle mappe, non sull'intera immagine di partenza. Il vantaggio delle CNN è dato dal fatto che le features sono estratte da pixel adiacenti, mentre nelle reti neurali classiche le features che si riferiscono a pixel distanti hanno la stessa importanza delle caratteristiche locali a causa della piena connessione di tali reti [53].

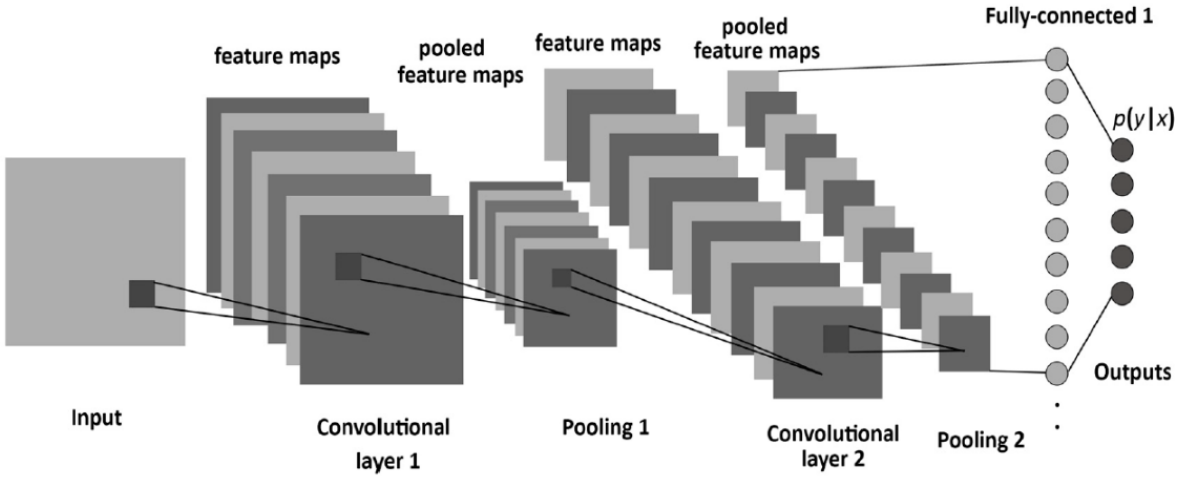

Figura 19: Tipica architettura di una CNN [54]

Dal momento che spesso il numero di pesi che devono essere allenati è relativamente alto se confrontato con la quantità di campioni di dati a disposizione per la fase di addestramento, non è infrequente che le CNN siano soggette al fenomeno chiamato overfitting, la cui insorgenza sta a significare che il modello apprende in modo eccellente i dati usati per l'addestramento ma va incontro a prestazioni deludenti se testato su dati non utilizzati in precedenza. Inoltre, le CNN possono non convergere qualora il set di dati usato per il training sia di dimensioni ridotte, non riuscendo ad apprendere alcuna relazione e fallendo nella fase di addestramento. Una tecnica utilizzata per migliorare le capacità di apprendimento in caso di dataset di dimensioni ridotte prende il nome di transfer learning [55]. Tramite questa tecnica, non è più necessario addestrare il proprio modello da zero, con una inizializzazione dei pesi arbitraria, ma è possibile perfezionare l'addestramento (fine tuning) a partire da pesi pre-addestrati, per un problema simile, per mezzo di un dataset di grandi dimensioni.

Un'altra tecnica utilizzata per migliorare le capacità delle reti neurali di generalizzare è stata introdotta da Hinton et al. [56] e prende il nome di dropout. In questo caso, i pesi dei neuroni possono essere impostati a zero o eliminati con una probabilità predefinita. Se il peso è zero significa che il neurone corrispondente non contribuisce all'output della rete e dunque non partecipa alle iterazioni dell'addestramento. Quindi, per ogni nuovo campione di training, l'architettura viene modificata eliminando casualmente nuovi set di neuroni.

Nei paragrafi seguenti sono descritte brevemente, in ordine di pubblicazione, le architetture delle diverse CNN selezionate per confrontare le rispettive abilità nell'estrazione di feature, a partire dalle immagini contenute nei dataset creati, al fine di classificare le diverse categorie di difetto.

4.2.1 VGG

VGG è l'acronimo di Visual Geometry Group [57], il dipartimento di scienze ingegneristiche dell'Università di Oxford che ha progettato le architetture di rete VGG-16 e VGG-19 [58] utilizzate nel presente lavoro. Si tratta di modelli adatti per i task di classificazione e detection che differiscono solamente per il numero di strati convoluzionali, rispettivamente 13 e 16, mentre entrambe dispongono di 3 strati densi per la classificazione. Spesso la VGG-16 è più utilizzata della VGG-19, data la minore complessità e la capacità di fornire prestazioni paragonabili.

I due modelli richiedono immagini di input di dimensioni 224x224x3, consentendo di processare immagini a colori (tre canali, RGB).

L'architettura fa uso di Kernel convoluzionali di dimensione 3x3 e Kernel per il Max pooling di dimesione 2x2. Gli sviluppatori delle reti VGG affermano che la combinazione in pila di tre filtri 3x3 ha lo stesso campo ricettivo di un singolo filtro convoluzionale di dimesione 7x7. L'utilizzo di filtri di dimensione ridotta comporta due vantaggi principali: il primo riguarda il numero di pesi da addestrare, significativamente ridotti nella versione in pila; il secondo riguarda la possibilità di applicare una funzione di attivazione dopo ogni filtro della pila. Pertanto, con questa configurazione è possibile estrarre features più complesse.

Le architetture delle VGG-16 e VGG-19 sono illustrate rispettivamente in Figura 20 e 21.

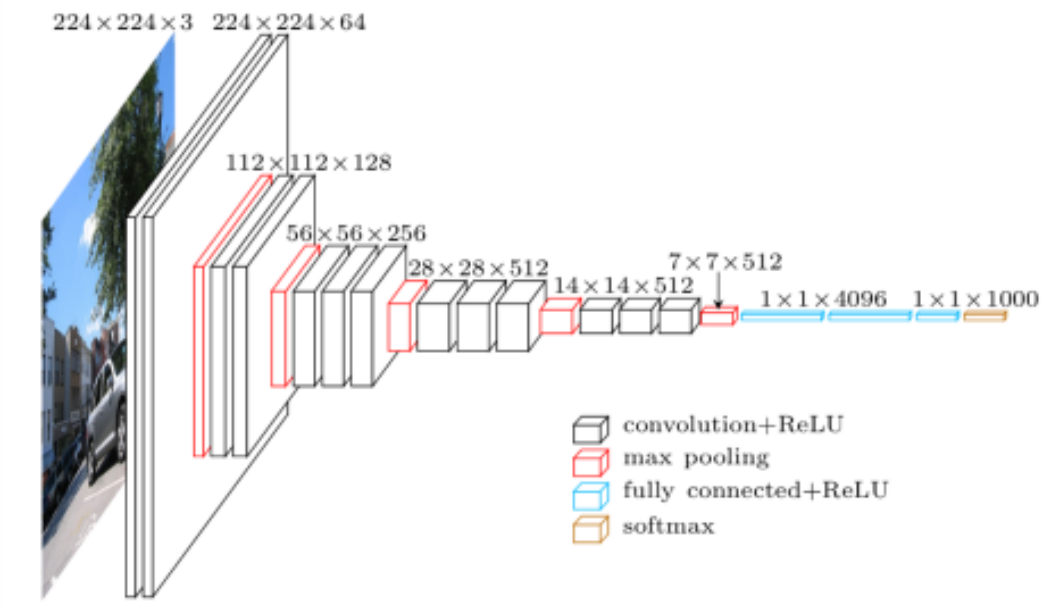

Figura 20: Architettura VGG-16 [59]

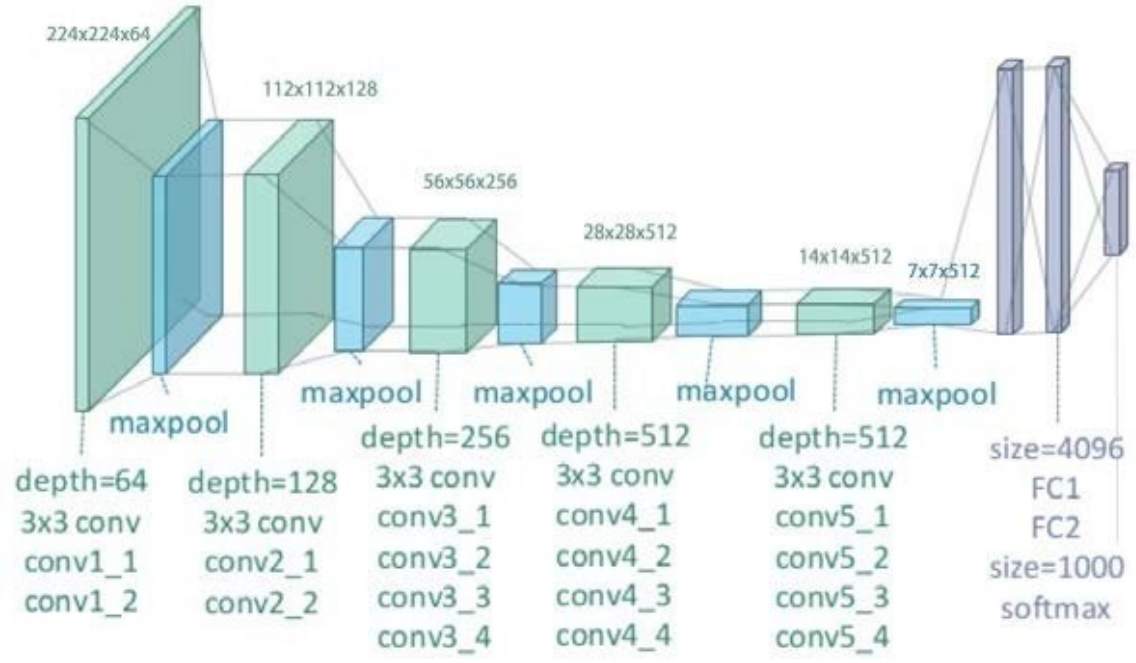

Figura 21: Architettura VGG-19 [60]

4.2.2 ResNet

Un altro modello di rete molto utilizzato per task di computer vision è stato proposto per la prima volta da Kaiming He et al. ed è denominato ResNet [61], abbreviazione di "Residual Network". Questo modello ha riscosso da subito un notevole successo, avendo raggiunto nel suo insieme la prima posizione nella competizione di classificazione ILSVRC 2015 [62] con solo il 3,57% di errore sulle cinque previsioni, oltre ad aver primeggiato in altre importanti competizioni dell'anno in cui stato presentato. ResNet è presente con numerose varianti che condividono lo stesso metodo di funzionamento, ciò che cambia è il diverso numero di livelli. Nel presente lavoro sono state testate tre differenti architetture, vale a dire ResNet50, ResNet101 e ResNet152, ad indicare rispettivamente 50, 101 e 152 livelli. Il concetto alla base di questi modelli, come suggerito dal nome esteso, è l'utilizzo della tecnica nota come skip connection. In sintesi, i dati elaborati, trasferiti normalmente al livello successivo, sono aggiunti anche all'output di quelli precedenti, tramite connessioni dirette tra strati convoluzionali non adiacenti. In questo modo si va a risolvere uno dei problemi che accomuna molte reti neurali profonde (DNN), riassunto da due concetti: vanishing e exploding gradient. Normalmente, nelle DNN è applicato l'algoritmo di apprendimento definito di backpropagation, coadiuvato da un metodo di ottimizzazione, come ad esempio lo Stochastic Gradient Descent (SGD), il cui scopo è trovare il punto di minimo globale di una funzione costo, che sta ad indicare quanto la previsione della rete differisca da quella reale. Negli approcci classici, trovato il gradiente della funzione costo, si aggiornano i parametri della rete, vale a dire i pesi, propagando all'indietro l'errore calcolato. Tuttavia, con l'aumento del numero di livelli, il gradiente può diventare molto grande (exploding) causando problemi di instabilità e generando parametri che superano quelli gestibili dall'elaboratore, o molto piccolo (vanishing), con conseguente aggiornamento minimo dei pesi e rallentamento dell'addestramento. Fatta questa premessa, è possibile comprendere meglio il funzionamento delle skip connection, tramite le quali al gradiente, invece di aspettare che si propaghi all'indietro tramite backpropagation, è consentito di raggiungere i nodi iniziali saltando quelli intermedi, in modo da apprendere molti più parametri in maniera efficace. In Figura 22 è riportata la sola architettura della ResNet34, non utilizzata nel presente lavoro ma utile per capire la struttura di base del modello generale e più semplice da rappresentare. Per questioni di praticità la rete è stata spezzata in due e lo strato precedente e successivo del punto di rottura rappresentano lo stesso livello.

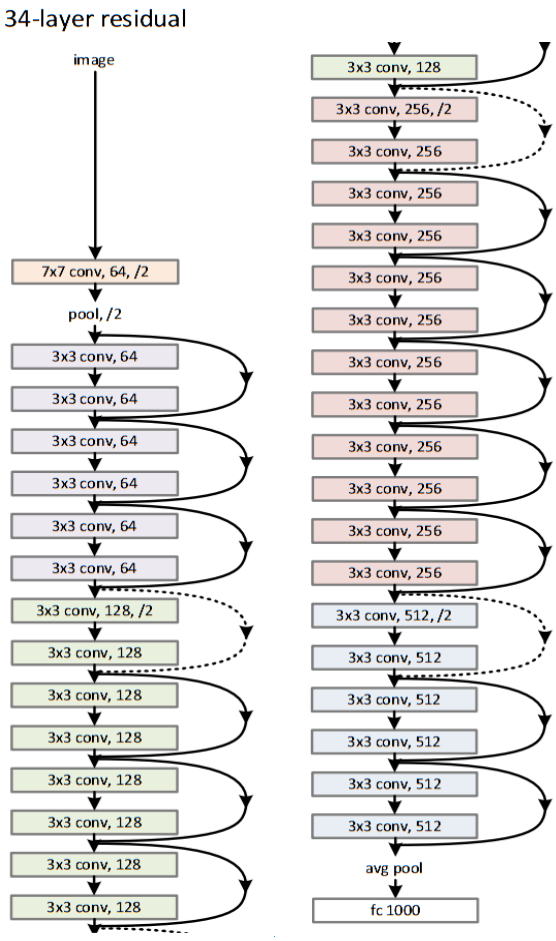

Figura 22: Architettura ResNet34 [61]

4.2.3 Inception

La rete Inception è stata presentata nella sua prima versione da Szegedy et al. [63] e successivamente migliorata e proposta in diverse altre pubblicazioni. Nel presente lavoro è stata utilizzata la terza versione, nota come Inception V3 [64]. A differenza di molte delle architetture più popolari, in cui si cercano prestazioni migliori semplicemente aggiungendo in pila strati convoluzionali sempre più profondi, la Inception presenta una struttura più complessa utilizzando Kernel di dimensioni diverse per lo stesso livello, con l'obiettivo di migliorare velocità e accuratezza. Spesso la sede dell'informazione presente nelle immagini è soggetta a grandi variazioni, rendendo difficoltosa la scelta della corretta dimensione del Kernel per le operazioni di convoluzione. Infatti, è preferibile un Kernel di dimensione maggiore nel caso di informazione distribuita più globalmente, mentre uno di dimensione minore è la scelta migliore quando l'informazione è distribuita localmente. Inoltre, reti molto profonde sono soggette a overfitting e hanno elevati costi

computazionali. Per risolvere queste problematiche vengono utilizzati più Kernel di dimensioni diverse per ogni strato, le cui uscite vengono concatenate e inviate al modulo successivo. Nella terza versione vengono proposti diversi accorgimenti per migliorare i modelli precedenti, tra i quali l'uso del "label smoothing", vale a dire un tipo di componente di regolarizzazione aggiunto alla formula della loss che impedisce alla rete di diventare troppo sicura su una classe, prevenendo l'overfitting, convoluzioni fattorizzate 7x7 e l'uso di un classificatore ausiliario per propagare le informazioni etichettate negli strati più profondi della rete, unito ad una batch normalization.

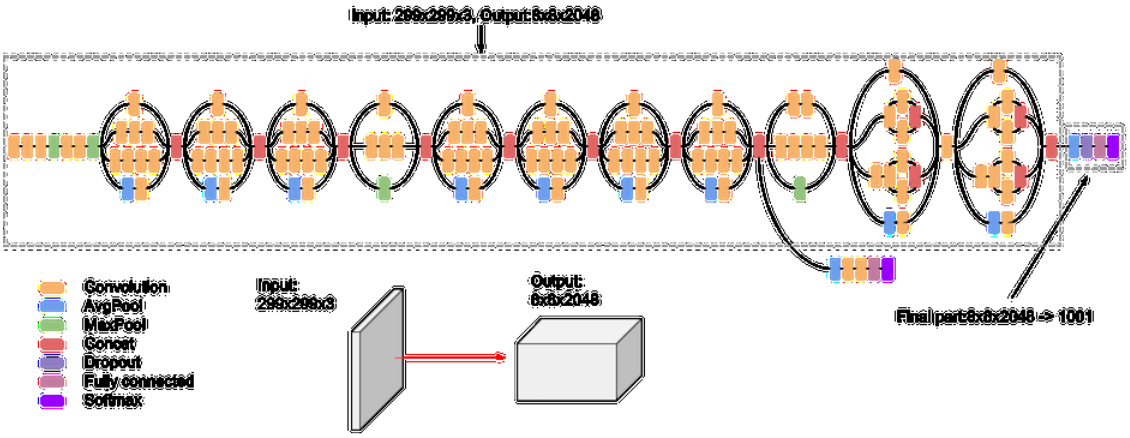

Figura 23: Architettura Inception V3 [64]

4.2.4 DenseNet

Nelle classiche reti neurali convoluzionali, l'immagine di ingresso subisce numerose operazioni di convoluzione in serie, in modo da estrarre feature di alto livello. Come illustrato in precedenza, con l'avvento delle ResNet si è introdotto il concetto delle skip connection, che permettono il trasferimento dell'informazione anche a livelli non adiacenti. Huang et al. [65] propongono un nuovo modello di rete neurale chiamato DenseNet. Questa tecnica si basa sul concetto della concatenazione, in cui ogni strato riceve ingressi addizionali da quelli che lo precedono, e trasferisce le proprie mappe di feature a tutti gli strati successivi. Ciò significa che l'ultimo strato di output ha ottenuto informazioni dirette da ogni strato che lo ha preceduto, incluso quello di ingresso. In questo modo, la rete può risultare più compatta e con un minor numero di canali, comportando una maggiore efficienza computazionale e di risorse. Una DenseNet è costituita da strati di composizione, bottleneck e transizione. La struttura di base della DenseNet è caratterizzata

dall'avere Batch Normalization, funzione di attivazione ReLU e filtri 3x3 per ogni strato di composizione. Per ridurre dimensione e complessità del modello, prima della precedente viene inserita una combinazione dello stesso tipo ma con filtri 1x1. Per gli strati di transizione tra due blocchi densi contigui, sono utilizzati filtri 1x1 seguiti da average pooling di dimensione 2x2. Le mappe di feature hanno la stessa dimensione all'interno dei blocchi densi in modo da poter essere facilmente concatenate. Alla fine dell'ultimo blocco viene effettuato un global average pooling, seguito da un classificatore con funzione di attivazione softmax. Nell'immagine sottostante è rappresentato un esempio dell'architettura di una DenseNet.

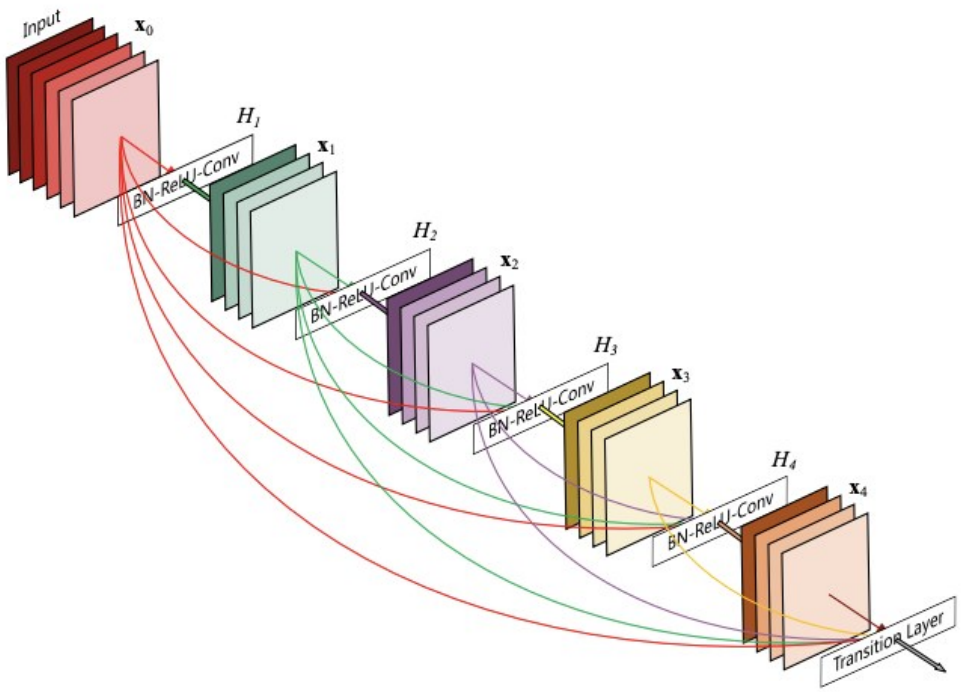

Figura 24: Architettura DenseNet [65]

4.2.5 MobileNet

MobileNet [66] è una rete neurale presentata da Google per cercare di adattare la rete Inception per dispositivi portatili e applicazioni di visione embedded. Questo modello è basato su una architettura che utilizza "convoluzioni separabili profonde" (depthwise separable convolutions) per realizzare reti neurali profonde più leggere. L'obiettivo dei ricercatori era di ridurre il numero di parametri e calcoli, preservando il più possibile le prestazioni. L'architettura su cui è basata ha come idea principale di avere filtri diversi per differenti canali di input. Si tratta di una forma di

convoluzioni che fattorizzano una convoluzione standard in una profonda e una 1x1 chiamata convoluzione punto per punto (pointwise convolution). Per la MobileNet la convoluzione depthwise applica un singolo filtro per ogni canale di input. In seguito, la convoluzione pointwise applica una convoluzione 1x1 per unire le uscite della depthwise che la precede. A differenza di una convoluzione classica in cui filtraggio e unione degli input avvengono in un unico step per generare un nuovo insieme di output, nella depthwise queste operazioni sono separate in due strati distinti, uno per il filtraggio e uno per l'unione. Questa fattorizzazione ha l'effetto di ridurre drasticamente il calcolo e le dimensioni del modello. Infatti, la rete MobileNet ha ridotto significativamente il numero dei parametri e, di conseguenza, il calcolo richiesto è in genere solo 1/9 della rete originale, con una piccola riduzione della precisione. La rete è costituita dalle convoluzioni appena descritte eccetto che per il primo strato, dato da una full convolution. Ad ogni strato segue una BatchNormalization e funzione di attivazione ReLU, ad eccezione dell'ultimo strato denso associato ad una softmax per la classificazione. Prima dello strato fully connected finale la riduzione spaziale viene ridotta ad 1 tramite un average pooling. Considerando le convoluzioni depthwise e pointwise separate, la rete MobileNet è costituita da 28 strati.

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

Figura 25: Architettura MobileNet [66]

4.2.6 Xception

Il modello denominato Xception [67] rappresenta la versione “estrema” della rete Inception descritta in precedenza. Basata anch’essa su convoluzioni separabili profonde, ha mostrato risultati migliori della Inception V3 per entrambi i dataset ImageNet ILSVRC e JFT. Come riportato in Figura 26, la rete Xception può essere suddivisa in tre parti principali: Entry flow, Middle flow, ripetuto per otto volte, e Exit flow.

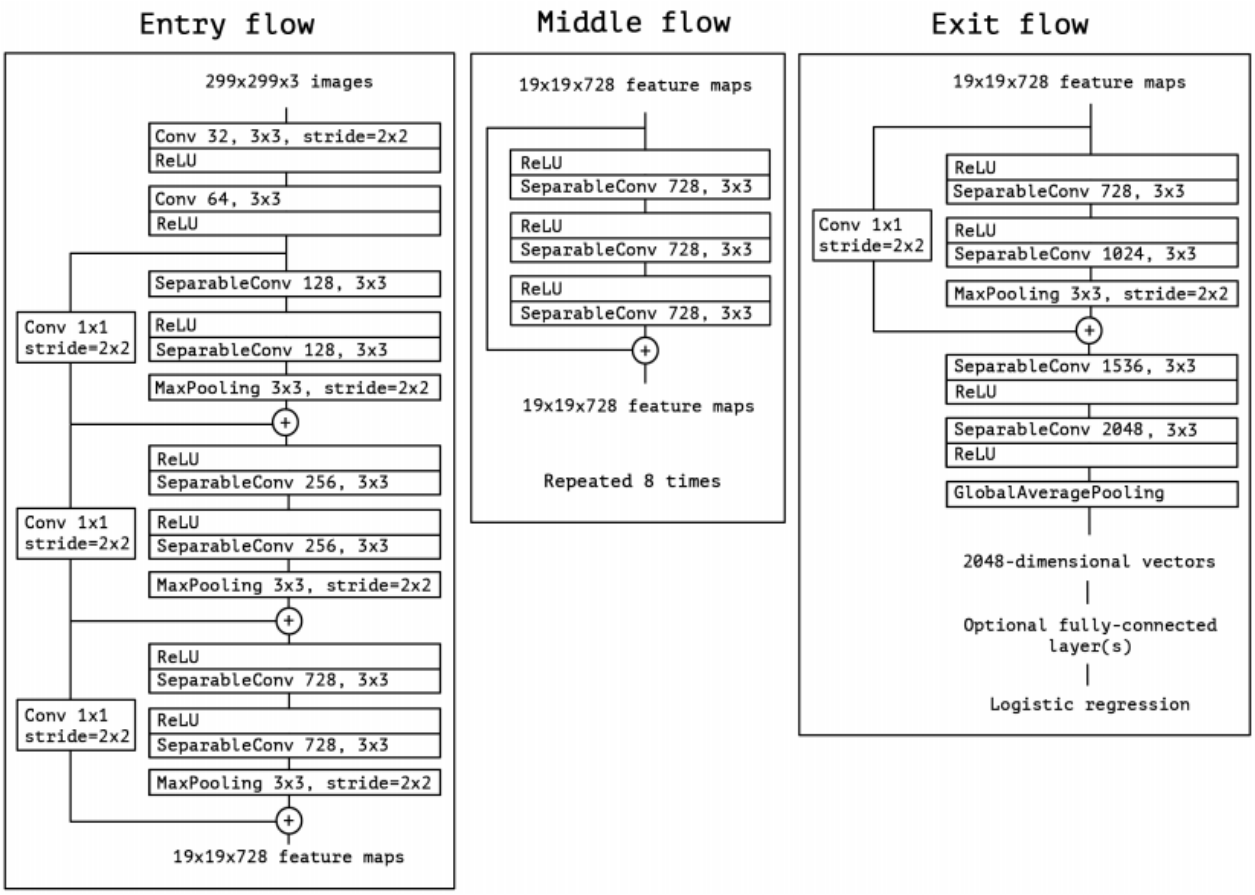

Figura 26: Architettura Xception [67]

Dalla figura risulta evidente la struttura basata su convoluzioni separabili profonde, modificata rispetto alla versione classica presentata precedentemente per la MobileNet. Nella versione originale, la convoluzione separabile profonda non è altro che la convoluzione depthwise, vale a dire la convoluzione spaziale nxn per canale, seguita da una convoluzione pointwise, convoluzione 1x1 per cambiare dimensione. Nella versione utilizzata dalla Xception, invece, la convoluzione separabile è invertita, essendo costituita prima dalla pointwise, seguita poi dalla depthwise. Questa modifica deriva dalla struttura della rete Inception V3, in cui la convoluzione 1x1 precede ogni

convoluzione spaziale nxn. Uno dei vantaggi che questo approccio apporta rispetto agli strati convoluzionali tradizionali consiste nella favorevole dimensione dei filtri, risultando vantaggioso sia in termini computazionali che di occupazione di memoria. La principale differenza deriva dal fatto che per convoluzioni tradizionali si trasforma l'immagine numerose volte, mentre per le separabili ciò avviene in un'unica operazione.

4.2.7 NASNet

NASNet [68] è un modello di rete nuovamente presentato da Google, che basa il suo funzionamento sul cercare la migliore combinazione di parametri del dato spazio di ricerca di dimensioni del filtro, canali di output, strides, numero di livelli, ecc. In questa impostazione di apprendimento per rinforzo, la ricompensa dopo ogni azione di ricerca è data dall'accuratezza per l'architettura cercata sul dataset dato.

In questo caso, il metodo di ricerca utilizzato è denominato Neural Architecture Search (NAS). Una rete neurale ricorrente, chiamata controllore, campiona reti figlie con architetture differenti. Queste ultime sono addestrate a convergere per ottenere determinati livelli di accuratezza su un set di validazione. Le accuratezze così determinate vengono poi utilizzate per aggiornare la RNN in modo che sia in grado di generare architetture migliori nel tempo. Inizialmente, i ricercatori hanno testato il modello sul dataset CIFAR-10 per cercare il migliore strato convoluzionale o cella, per poi applicarla al dataset ImageNet mettendo in pila più copie di queste celle. In altri termini, rispetto a tutte le reti analizzate fino ad ora, in cui i blocchi convoluzionali sono definiti dagli autori, in questo caso, sebbene la struttura generale della rete sia predefinita, i blocchi o le celle non lo sono, ma vengono cercate tramite la tecnica del reinforcement learning. In Figura 27 è possibile osservare l'architettura del modello del controllore che costruisce in maniera ricorsiva un blocco di una cella convoluzionale.

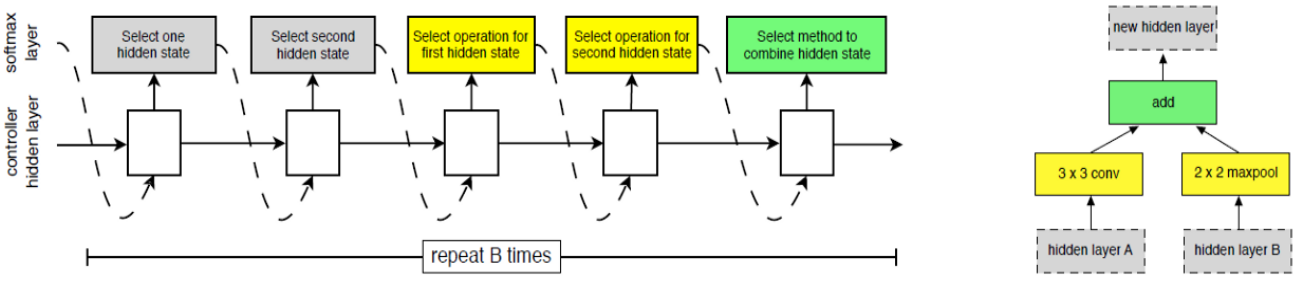

Figura 27: NASNet – controllore [68]

4.3 Tecnologie utilizzate

In questo paragrafo verranno brevemente presentate le tecnologie che hanno permesso la scrittura del codice e l'esecuzione dei test. Nello specifico, sarà descritto l'ambiente di sviluppo Google Colab e introdotto il linguaggio Python, che mettono a disposizione numerose librerie e plug-in utili per sviluppare le reti neurali convoluzionali utilizzate e algoritmi di intelligenza artificiale in genere.

4.3.1 Google Colab

Google Colab [69] è uno strumento gratuito presente nella suite di Google che consente di scrivere codice Python direttamente dal proprio browser. Una piattaforma online che offre un servizio di cloud hosting per notebook Jupyter dove creare documenti che contengono righe di codice, grafici, testi, link e molto altro. Si può creare un documento Google Colab direttamente da Google Drive e, proprio come un qualunque documento in G Suite, può essere condiviso con altri utenti che hanno la possibilità di modificarlo e lasciare commenti direttamente nel notebook. Il notebook Jupiter verrà poi eseguito su macchine virtuali di server Google. Ciò consente di svincolarsi dalla parte hardware e di concentrarsi esclusivamente sul codice Python e sui contenuti che si vuole integrare nel notebook. In Google Colab le risorse che vengono messe a disposizione agli utenti sono limitate, e variano a seconda delle fluttuazioni nella domanda. Se si ha bisogno di maggior stabilità, di macchine più performanti o di accedere a GPU e TPU più potenti, è possibile utilizzare la licenza a pagamento presente nella versione Pro di Google Colab. Normalmente, per l'utilizzo tramite licenza gratuita vengono messe a disposizione GPU K80, il tempo a disposizione è limitato a 12 ore, i timeout per inattività sono molto rigidi e si ha accesso a RAM al più di 13 Gb. Sono disponibili due versioni a pagamento, denominate Colab Pro e Colab Pro+. Nel presente lavoro, per svolgere gran parte dei test si è utilizzata la versione Pro, tramite la quale i blocchi note possono rimanere connessi fino a 24 ore, si hanno a disposizione RAM dalla capacità doppia e accesso a GPU T4 e P100.

4.3.2 Python

Come linguaggio di programmazione per scrivere le parti di codice relative alla creazione dei modelli e all'esecuzione dei test è stato scelto Python [70], per via della sua semplicità e per il fatto che è nativamente supportato da Google Colab. Python è un linguaggio di programmazione ad alto livello, rilasciato pubblicamente per la prima volta nel 1991 dal suo creatore Guido van Rossum.

Attualmente, lo sviluppo di Python, grazie all'enorme e dinamica comunità internazionale di sviluppatori, viene gestito dall'organizzazione no-profit Python Software Foundation. Python supporta diversi paradigmi di programmazione, come object-oriented, imperativo e funzionale.
È fornito di una libreria built-in estremamente ricca che, unitamente alla gestione automatica della memoria e a robusti costrutti per la gestione delle eccezioni, ne fa uno dei linguaggi più completi e comodi da usare. Si tratta di un linguaggio interpretato: un interprete si occupa di analizzare il codice sorgente (semplici file testuali con estensione .py) e, se sintatticamente corretto, di eseguirlo. In Python, non esiste una fase di compilazione separata (come avviene in C, ad esempio) che generi un file eseguibile partendo dal sorgente. L'essere interpretato lo rende un linguaggio portabile. Una volta scritto un sorgente, può essere interpretato ed eseguito sulla gran parte delle piattaforme utilizzate, siano esse di casa Apple (Mac) che PC (Microsoft Windows e GNU/Linux). Python mette a disposizione una delle librerie open source più famose in ambito deep learning, vale a dire Tensorflow [71], tramite cui è possibile velocizzare significativamente i calcoli dei diversi algoritmi testati grazie a una serie di funzioni che sono in grado di utilizzare i tensori e alla possibilità di sfruttare il calcolo parallelo fornito dalle GPU. Mette a disposizione inoltre un'altrettanto famosa API chiamata Keras [72], molto utilizzata per creare algoritmi di machine e deep learning con poche righe di codice, disponendo di molti dei più famosi modelli che possono essere facilmente importati e modificati. Tra le altre librerie open source per il linguaggio Python che consentono di effettuare numerose operazioni in ambito deep learning, e largamente utilizzate anche per il presente scopo, si possono citare Scikit-learn [73], NumPy [74], Matplotlib [75], SciPy [76] e OpenCV [77].

5 Esperimenti

In questo capitolo vengono riportati i passaggi fondamentali che hanno permesso l'esecuzione dei vari esperimenti compiuti a partire dai dataset presentati nel precedente capitolo. Saranno per prima cosa descritte le azioni preliminari alla fase di addestramento dei modelli proposti e verrà quindi illustrata la pipeline di preprocessing del dataset, riportando snippet di codice usato per lo scopo. In seguito, dopo aver elencato i principali criteri di valutazione adottati e descritto le modalità di svolgimento dei test, verranno riportati i risultati relativi alle diverse reti testate sui quattro dataset. Tutti i test sono svolti in ambiente Google Colab, inizialmente tramite licenza gratuita e in seguito tramite la versione pro per i test che necessitavano di maggiore potenza computazionale.

5.1 Preparazione del dataset

Prima di ogni esperimento è prevista una fase in cui si prepara il dataset per poter essere utilizzato dai modelli di rete per la successiva classificazione. La prima azione da fare consiste nel caricare il dataset nel proprio ambiente Google Drive per la successiva gestione dei dati direttamente dal notebook Jupyter messo a disposizione da Colab. In seguito, si definiscono delle utili funzioni per manipolare le immagini contenute nel dataset, tramite operazioni di data augmentation, gestione del colore e dell'illuminazione. Infine, le immagini vengono etichettate seguendo la distinzione data dalla classe di appartenenza, per poter essere facilmente utilizzate dai modelli seguenti.

5.1.1 Caricamento del dataset

Nel presente paragrafo verranno descritti i passaggi compiuti dall'avvenuta creazione del dataset al suo utilizzo all'interno del notebook. Esistono molti modi per accedere a file sul proprio computer o dalla propria cartella Drive e in questo caso verrà riportato il metodo seguito per tutti gli esperimenti. La prima azione ha riguardato la conversione del dataset in un archivio .zip e il caricamento nella propria cartella di lavoro in Drive. Per poter effettuare l'upload nella propria pagina Colab, si va ad ottenere dal file appena caricato un identificativo cliccandovi con il tasto destro e seguendo il percorso "ottieni link -> copia link". Ciò che si ottiene è un link condivisibile, in cui è presente l'identificativo cercato che verrà poi copiato in un apposito campo all'interno del codice per caricare l'oggetto che rappresenta il dataset.

A titolo di esempio si riporta il link ottenuto da uno dei dataset realizzati e caricati nella cartella Drive, con evidenziato l'id da copiare:

https://drive.google.com/file/d/15GMQVpidStDT0UN_0JKwE0EF6nRj9LzJ/view?usp=sharing

A questo punto è possibile passare al proprio notebook Colab in cui, dopo una cella di codice in cui si effettua l'autenticazione, necessaria per poter accedere ai file presenti in Drive, è possibile copiare l'id ottenuto in precedenza e importare il dataset di interesse, come mostrato nella cella di codice seguente:

```
# Original dataset uploading and unzip (multi class)
fileId = drive.CreateFile({'id': '15GMQVpidStDT0UN_0JKwE0EF6nRj9LzJ'})  # id of the dataset
fileId.GetContentFile(fileId['title'])  # Save Drive file as a local file
!unzip -q carbon_multi_class_2.zip
```

5.1.2 Data augmentation

Come accennato nel precedente capitolo parlando della costruzione dei tre differenti dataset, per ciascuno di essi è stata prevista una funzione che effettuasse la data augmentation tramite flip orizzontale e verticale delle immagini di partenza. La data augmentation è importante nel caso in cui i dati siano limitati in numero, poiché consente di evitare di andare incontro ad overfit, andando ad aumentare in maniera fittizia la dimensione del dataset. Normalmente la data augmentation viene effettuata solamente sul training set, per evitare di generare l'effetto opposto al desiderato, vale a dire favorire l'overfit. Inoltre, deve prevedere azioni che siano coerenti con la natura del dato a disposizione. Nel caso in esame, data la struttura dei fasci di fibre di carbonio che si ripete periodicamente, dei difetti che possono insorgere in qualsiasi punto della superficie e dal fatto che questi ultimi possono essere individuati da qualunque posizione si osservi il pezzo, si è ritenuto di effettuare le due operazioni di cui sopra non solo per il training set, ma anche per le restanti immagini del dataset, creandone uno nuovo con immagini aumentate per ciascuna presente in quelli di partenza. In altre parole, si è considerato come qualunque immagine generata artificiosamente potesse essere stata acquisita direttamente dal pezzo da esaminare, per essere poi inserita nel dataset originale senza per questo essere considerata "fittizia". Per concludere, nel presente contesto l'utilizzo di tali tecniche ha consentito di velocizzare la costruzione di un dataset di dimensioni maggiori, poiché le stesse immagini aumentate sarebbero state in alternativa

direttamente acquisite sul campo. Nel seguito viene riportata la funzione utilizzata per lo scopo, nel caso specifico relativo alla seconda versione del dataset a tre classi:

```
def data_augmentation(dataset_path, augmented_dataset_path):

    """ Data augmentation function for horizontal and vertical flip

    Parameters
    ----------
    dataset_path : str
                   Path to the folder with images to be augmented
    augmented_dataset_path : str
                             Destination path for the synthetic samples. It will
                             contain the same folders which are in dataset_path,
                             with new data
    """

    path = dataset_path

    if not os.path.exists(augmented_dataset_path):
        os.makedirs(augmented_dataset_path)

    for defect_type in os.listdir(path):
        i = 500
        if not os.path.exists(augmented_dataset_path + '/' + defect_type):
            os.makedirs(augmented_dataset_path + '/' + defect_type)
            read_path = path + '/' + defect_type + '/'
            write_path = augmented_dataset_path + '/' + defect_type + '/'
        for files in os.listdir(read_path):
            img_to_augment = cv2.imread(read_path + files)
            horizontal_img = cv2.flip(img_to_augment, 0)
            vertical_img = cv2.flip(img_to_augment, 1)
            root, ext = os.path.splitext(files)
            if defect_type == 'negative':
                cv2.imwrite(os.path.join(write_path, 'negative_' + str(i + 1) + ext), horizontal_img)
                cv2.imwrite(os.path.join(write_path, 'negative_' + str(i + 2) + ext), vertical_img)
            elif defect_type == 'non_recoverable_defects':
                cv2.imwrite(os.path.join(write_path, 'nrd_' + str(i + 1) + ext), horizontal_img)
                cv2.imwrite(os.path.join(write_path, 'nrd_' + str(i + 2) + ext), vertical_img)
            elif defect_type == 'recoverable_defects':
                cv2.imwrite(os.path.join(write_path, 'rd_' + str(i + 1) + ext), horizontal_img)
                cv2.imwrite(os.path.join(write_path, 'rd_' + str(i + 2) + ext), vertical_img)
            i += 2
```

Come è possibile osservare, la funzione accetta in ingresso due parametri, “dataset_path” ad indicare il percorso del dataset originale precedentemente caricato, e “augmented_dataset_path” per il dataset aumentato, distinto dal precedente, in cui verranno salvate le nuove immagini. Per ogni immagine contenuta nel percorso originale, si effettuano i flip tramite le funzioni messe a disposizione dalla libreria OpenCV, successivamente si vanno ad aggiungere nel nuovo dataset, denominandole con la dicitura utilizzata per le rispettive classi di appartenenza e numerandole in maniera incrementale.

Le nuove immagini non vengono salvate in coda al dataset originale ma direttamente nella versione aumentata, in quanto si è deciso di mantenere i due dataset separati in modo da avere sempre a disposizione la versione di partenza.
Infine, per trasferire al nuovo dataset anche le immagini originali, si utilizza un'altra funzione con il compito di copiarle. In questo modo si è ottenuto un nuovo dataset completo che è possibile scaricare e, seguendo la procedura descritta nel paragrafo precedente, spostarlo nel proprio Drive per poterlo impiegare allo stesso modo del dataset originale, evitando di ripetere la data augmentation ad ogni nuovo utilizzo del notebook.

5.1.3 Canali e illuminazione

Come verrà descritto nel seguito, al fine di trovare il modello di rete che fornisse le migliori prestazioni per il task di classificazione dei difetti, per ciascuna delle reti più promettenti sono state provate diverse configurazioni, sia in termini di iper-parametri, sia per la tipologia di immagini passate in ingresso. I dataset realizzati sono costituiti da immagini RGB, dunque a tre canali di colore. Per verificare se ciò influisse in maniera significativa sulle prestazioni finali, è stata realizzata una semplice funzione, di cui si riporta il breve codice, che consentisse di convertire le immagini di partenza in scala di grigi, permettendo di effettuare i test in entrambe le condizioni:

```
def convert_gray(path):

    """ Convert path images to grayscale

    Parameters
    ----------
    dataset_path : str
                   Path to the folder with images to be converted
    """
    for defect_type in os.listdir(path):
        read_path = path + '/' + defect_type + '/'
        for files in os.listdir(read_path):
            img = cv2.imread(read_path + files)
            gray = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
            cv2.imwrite(os.path.join(read_path + files), gray)
```

In aggiunta alla variazione di colore, è stata realizzata un'altra funzione che consentisse di variare le condizioni di luce delle immagini. Dalle ricerche in letteratura, infatti, è emerso come nella maggior parte dei casi si effettuasse una normalizzazione dell'illuminazione nella fase di preprocessing, in

particolare facendo uso del filtro CLAHE (Contrast Limited Adaptive Histogram Equalization) [78]. Si tratta di una tecnica di elaborazione digitale delle immagini usata per migliorarne il contrasto. Si differenzia dal metodo ordinario di equalizzazione dell'istogramma in quanto calcola diversi istogrammi, ognuno corrispondente a una distinta sezione dell'immagine, e li usa per distribuire i valori di luminosità. Anche in questo caso, sia la conversione in scala di grigi che l'applicazione del filtro CLAHE sono state implementate grazie alle funzionalità della libreria OpenCV. Di seguito si riporta la sezione di codice corrispondente al filtraggio:

```
def illumination_normalization(path):

    """ Function for Local Illumination Normalization (CLAHE)

    Parameters
    ----------
    path : str
           Path to the folder with images to be augmented
    """
    for defect_type in os.listdir(path):
        read_path = path + '/' + defect_type + '/'
        for files in os.listdir(read_path):
            img = cv2.imread(read_path + files, 0)
            clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))
            new_img = clahe.apply(img)
            cv2.imwrite(os.path.join(read_path + files), new_img)
```

5.1.4 Etichettatura

Al termine delle operazioni da effettuare per la fase di preprocessing si trova la parte relativa al labelling dei campioni da utilizzare nel seguito. Questa fase avviene nello stesso modo indipendentemente dalla natura delle immagini ottenute giunti a questo punto, che siano RGB, in scala di grigi e/o con applicate tecniche di filtraggio. L'unica differenza è riscontrabile tra dataset binario e multi-classe, banalmente nel nome della classe da cui prelevare le immagini e dal numero arbitrario assegnato, ad indicare le immagini appartenenti alla stessa classe. Prima di effettuare il labelling vengono lette le immagini, dalla cartella in cui sono salvate, con una target size di 224x224 o 299x299, in base al modello di rete che si intende utilizzare. Questa operazione viene fatta poiché l'obiettivo è di eseguire l'addestramento a partire da modelli pre-allenati, dunque eseguire un fine-tuning, e per farlo è necessario fornire in ingresso immagini della stessa dimensione di quelle utilizzate per pre-addestrare la rete. Dopo aver letto l'immagine la si trasforma in un array e si

effettua il reshape. Successivamente, le varie immagini vengono etichettate in base alla classe di appartenenza. Per completezza, di seguito viene riportato il metodo utilizzato per il labelling nel caso di dataset a tre classi così come proposto nel presente lavoro:

```
for defect_type in os.listdir(path):
    read_path = path + '/' + defect_type + '/'
    for filename in os.listdir(read_path):
        if ConvModel == XceptionModel or ConvModel == InceptionV3Model:
            imgs = load_img(os.path.join(read_path, filename), target_size=(299,299))
        else:
            imgs = load_img(os.path.join(read_path, filename), target_size=(224,224))
        imgs = img_to_array(imgs)
        imgs = imgs.reshape((1, imgs.shape[0], imgs.shape[1], imgs.shape[2]))
        imgs = preprocess_input(imgs)
        imgs_array[index,:,:,:] = imgs
        if defect_type == 'negative':
            label_array.append(0)
        elif defect_type == 'non_recoverable_defects':
            label_array.append(1)
        elif defect_type == 'recoverable_defects':
            label_array.append(2)
        index += 1
```

5.2 Criteri di valutazione

Per valutare le prestazioni delle CNN testate sono stati presi in considerazione i seguenti criteri:

- **Accuratezza**

  Rappresenta il criterio basilare per valutare la correttezza della classificazione ed è definita come il rapporto tra il numero di predizioni corrette e il numero totale di predizioni. La formula che la rappresenta è definita come:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

  Dove TP rappresenta i veri positivi (True Positive), TN i veri negativi (True Negative), FP i falsi positivi (False Positive) e FN i falsi negativi (False Negative).

- **Precisione**

  La precisione è una metrica molto utilizzata in quanto è in grado di indicare quante volte il classificatore, tra tutti i campioni positivi identificati, è in grado di fornire la risposta corretta. È definita come il rapporto tra veri positivi e la somma tra veri positivi e falsi positivi:

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

- **Recall**

  La recall è un altro criterio molto utilizzato ed è definita come la frazione dei campioni di una classe che sono correttamente predetti dal modello. Più formalmente:

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

- **F1 – score**

  L'F1 - score è un criterio di valutazione che combina precision e recall ed è spesso utilizzato per applicazioni in cui entrambe sono importanti. Rappresenta la media armonica di precisione e recall ed è definita come:

$$F1 - score = \frac{2 * Precision * Recall}{Precision + Recall}$$

- **Matrice di confusione**

  La matrice di confusione è un utile criterio in quanto restituisce una rappresentazione dell'accuratezza della classificazione. Ogni colonna della matrice sta ad indicare i valori predetti, mentre ogni riga i valori reali. L'elemento C_{ij} di una matrice di confusione rappresenta quanti campioni della i - esima classe sono classificati come appartenenti alla j - esima classe. Nel caso di un classificatore ideale, la matrice di confusione è uguale alla matrice identità delle stesse dimensioni.

- **Curve ROC e AUC**

  La curva ROC (Receiver Operating Characteristic) è un grafico che mostra le performance di un classificatore, solitamente binario, come funzione della sua soglia di cut-off. In altre parole, viene creata tracciando il valore della frazione di veri positivi (True Positive Rate, TPR) rispetto alla frazione di falsi positivi (False Positive Rate, FPR) a varie impostazioni di soglia. Solitamente, in associazione alle curve ROC viene valutata anche la AUC (Area Under the Curve), che calcola l'area al di sotto della curva ROC e assume valori compresi tra 0 e 1. Un modo di interpretare l'AUC è come la probabilità che il modello classifichi un campione positivo casuale più in alto di un campione negativo casuale. In generale, più alto è il valore dell'AUC, migliore sarà il modello associato. Valori alti si raggiungono tanto più la parte centrale della curva ROC si avvicina all'angolo in alto a sinistra del grafico.

5.3 Descrizione degli esperimenti

Lo scopo degli esperimenti effettuati è di testare i diversi modelli presentati nel Capitolo 4, per verificare con quale rete e con quali parametri si ottengano le prestazioni migliori per il problema di classificazione di difetti estetici in materiali compositi. Durante il periodo di tirocinio e con la realizzazione nel tempo dei diversi dataset, sono stati effettuati numerosi esperimenti, variando spesso i parametri regolabili e apportando modifiche strutturali ai modelli. Le principali strade tentate hanno riguardato:

- modifiche nella fase di preprocessing;
- modifiche strutturali, con l'aggiunta di strati con il compito di classificare, posti a valle dei modelli impiegati per l'estrazione di features;
- tuning degli iper-parametri delle diverse architetture di rete.

Tra le modifiche nella fase di preprocessing rientrano le prove effettuate fornendo in ingresso alle reti immagini RGB o in scala di grigi, seguite o meno da una normalizzazione dell'illuminazione, come descritto nei paragrafi precedenti. Con le modifiche strutturali si racchiudono tutti i cambiamenti apportati direttamente alle reti, in particolare con l'aggiunta di differenti combinazioni di strati di pooling, BatchNormalization, Dropout e fully connected a valle dei modelli precostruiti, al fine di migliorare la capacità di classificazione. Dato il numero ridotto di immagini presenti nei dataset, si è

deciso di sfruttare la tecnica del transfer learning e utilizzare modelli pre-addestrati su dataset ImageNet [79], contenente milioni di immagini e ampiamente usato per applicazioni di classificazione o object detection. I dataset realizzati e costituiti da immagini di superfici di materiali compositi, invece, sono stati utilizzati per eseguire il fine-tuning delle reti pre-addestrate, andando a scongelare un numero differente di strati per ogni rete testata, in modo da determinare quanti livelli fosse conveniente scongelare per consentire l'aggiornamento dei pesi. A riguardo, sono stati fatti tentativi sia evitando di scongelare gli strati, mantenendo le reti con i pesi appresi nell'addestramento con ImageNet, sia scongelando da 4 a 16 strati, a seconda della rete testata. Come si avrà modo di constatare nel seguito, le prestazioni migliori sono state ottenute aggiornando i pesi degli ultimi 8 strati. Considerando il gran numero di combinazioni tentate, nel seguito verranno riportati i soli risultati più promettenti e le relative scelte effettuate a monte.
Altre modifiche hanno riguardato il tuning degli iper-parametri, vale a dire tutte le scelte che possono influire in maniera più o meno significativa nella performance del modello. Rientrano in questa categoria:

- dimensione delle batch con cui addestrare le reti;
- valore del random state per determinare i campioni da inserire nei set di training, validazione e test;
- numero di epoche e patience per l'early stopping;
- numero di split del dataset per ogni esperimento;
- ottimizzatore e learning rate utilizzati per la fase di addestramento.

In particolare, sono state provate batch costituite da 16 o 32 campioni, random state di valore pari a 0 o 42, epoche da 50 a 100 e funzione di early stopping con patience da 5 a 10, numero di split di valore 5 o 10, ottimizzatori SGD o Adam, entrambi con learning rate variabile da 0.1 a 0.0001.
Vale la pena sottolineare come per ciascun esperimento sia stata adottata la tecnica nota come stratified shuffle split tramite cui, prima di ogni iterazione della fase di apprendimento, il dataset viene rimescolato e i campioni in esso contenuti selezionati in maniera casuale per determinare i set di train, validation e test, cui spettano rispettivamente il 70%, 10% e 20% delle immagini del dataset completo, suddivisione ampiamente utilizzata in letteratura. La divisione tramite split è stata effettuata dalle 5 alle 10 volte per ogni modello testato sullo stesso dataset di partenza,

valutando le prestazioni della rete sulla media delle performance ottenute per i diversi split. In questo modo, è possibile valutare il comportamento del modello in questione in maniera più realistica rispetto al testarlo su un singolo split, avendo la possibilità di verificare se un comportamento eccessivamente buono della rete dipenda da uno split favorevole o se risulti stabile, mantenendosi sullo stesso range di valori per ogni split.

Come anticipato, alcune delle precedenti scelte hanno avuto influenza molto relativa o nulla, mentre altre hanno inciso in maniera notevole sulle prestazioni. Una analisi relativa alle scelte che hanno consentito di ottenere le performance migliori verrà fatta nel Capitolo 6.

Per supportare con maggiore concretezza i risultati ottenuti, si riportano in breve le versioni delle più importanti librerie utilizzate per ciascun esperimento e gli strumenti messi a disposizione dall'ambiente di lavoro Colab.

Tutti gli esperimenti sono stati eseguiti su Google Colab, utilizzando il runtime GPU e Keras 2.7.0, con il backend TensorFlow 2.7.0 e Scikit-learn 1.0.1.

Per l'addestramento dei modelli si è inizialmente utilizzata la versione gratuita messa a disposizione da Google Colab, tuttavia, a causa del maggior numero di immagini da processare presenti nei dataset a tre classi, si è ricorso alla versione pro. Ciò è dipeso dalla necessità di disporre di un maggior quantitativo di RAM, quasi sempre saturata nel caso di licenza gratuita. Di seguito vengono riportate le risorse messe a disposizione dall'ambiente Colab nella versione gratuita e pro relativa all'uso che ne ho fatto durante il periodo di tirocinio, precisazione doverosa poiché potrebbero risultare differenti nel tempo e in base al paese in cui si utilizza il servizio:

	Google Colab standard	Google Colab pro
CPU	Intel(R) Xeon(R) CPU @ 2.30GHz	Intel(R) Xeon(R) CPU @ 2.20/2.30 GHz
GPU	Tesla K80	Tesla T4 e Tesla P100
Memoria GPU	12 GB	16 GB
RAM	12.69 GB	25.46 GB
Disco	78.19 GB	166.83 GB
Durata VM	12 ore	24 ore

Tabella 2: Google Colab free vs Google Colab pro

5.4 Risultati

Nei successivi paragrafi saranno discussi i risultati raggiunti dai diversi modelli testati al variare delle condizioni descritte in precedenza, ponendo maggior enfasi sulle combinazioni che hanno fornito le prestazioni migliori. Si è deciso di strutturare la trattazione distinguendo nei due differenti problemi cui sono state sottoposte le reti, vale a dire di classificazione binaria e multi-classe. Si è iniziato testando il dataset binario e valutando le performance ottenute. Lo scopo dei primi esperimenti è di stimare la capacità delle architetture di CNN selezionate di distinguere tra immagini con difetti di trama e prive degli stessi. Alla luce dei risultati incoraggianti, si è proseguito testando gli stessi modelli con differenti combinazioni per il task multi-classe, raggiungendo in alcuni casi risultati migliori rispetto alla versione binaria. Nel secondo gruppo di esperimenti viene valutata la capacità dei modelli di distinguere tra immagini prive di difetti, con difetti recuperabili e con difetti non recuperabili, in cui rientrano le discontinuità di trama di cui sopra. Vale la pena ricordare come la maggior parte delle immagini siano state acquisite in condizioni non ottimali e l'insorgenza delle diverse tipologie di difetti può presentarsi in qualsiasi regione dell'immagine, condizioni che avrebbero potuto influire negativamente sulle prestazioni finali.

Per completezza, per ogni CNN testata, si riporta la versione implementata a partire dai modelli pre-addestrati disponibili in Keras:

- VGG16
- VGG19
- ResNet50V2
- ResNet101V2
- ResNet152V2
- InceptionV3
- DenseNet121
- MobileNetV2
- Xception
- NASNetMobile

5.4.1 Dataset binario

Una prima prova effettuata per il problema di classificazione binaria ha riguardato l'utilizzo sia del dataset originale, contenente 200 immagini per classe, sia della versione con aggiunte le immagini aumentate tramite flip orizzontale e verticale, costituita da 600 immagini per classe. Alla luce dei primi risultati ottenuti su alcuni dei modelli testati, si è notato come l'utilizzo di un maggior numero di immagini migliorasse notevolmente l'accuratezza nella classificazione e in generale gli altri criteri di valutazione considerati. Per questo motivo, si è deciso di proseguire i successivi test con la sola versione del dataset con immagini aumentate, scelta poi ripetuta per i test eseguiti per il task di classificazione a tre classi.

Le prestazioni migliori di classificazione binaria relative a ciascuna rete, per la versione aumentata del dataset, sono riportate in Tabella 3.

Per ogni esperimento sono state effettuate più prove con split differenti del dataset e i criteri di valutazione riportati rappresentano il valor medio dei valori raggiunti per i singoli split.

Architettura	Accuratezza	Loss
VGG16	0.940 +/- 0.015	0.191 +/- 0.040
VGG19	0.942 +/- 0.021	0.202 +/- 0.058
ResNet50V2	0.772 +/- 0.022	0.536 +/- 0.036
ResNet101V2	0.771 +/- 0.048	0.506 +/- 0.064
ResNet152V2	0.731 +/- 0.059	0.538 +/- 0.065
InceptionV3	0.731 +/- 0.027	0.526 +/- 0.032
DenseNet121	0.926 +/- 0.023	0.197 +/- 0.068
MobileNetV2	0.875 +/- 0.028	0.449 +/- 0.125
Xception	0.831 +/- 0.045	0.415 +/- 0.107
NASNetMobile	0.706 +/- 0.012	0.732 +/- 0.074

Tabella 3: Confronto delle performance per il task di classificazione binaria

Nella tabella precedente vengono riportati i nomi delle reti utilizzate per la fase di feature extraction, a cui fanno seguito i layer il cui scopo è effettuare la classificazione finale.

Dai risultati riportati in tabella risulta evidente come i modelli che raggiungono il livello di accuratezza maggiore nella classificazione siano appartenenti alla famiglia VGG, vale a dire VGG16 e VGG19. Un altro risultato degno di nota viene raggiunto dal modello DenseNet121, seguito dalla rete MobileNetV2. In tutti gli altri casi, i modelli implementati non hanno fornito prestazioni paragonabili o sufficientemente valide per possibili applicazioni a livello industriale.

Per favorire la comprensione dei risultati raggiunti e delle architetture implementate, si riportano nelle tabelle 4 - 6 i valori di Precisione, Recall e F1-score ottenuti per le due classi presenti nel dataset per le sole reti più promettenti, vale a dire VGG16, VGG19 e DenseNet121. Inoltre, nelle Figure 28 - 30 sono rappresentate le curve ROC con il rispettivo valore di AUC per i 10 split di ogni modello. Per concludere al termine verranno riportate le architetture e la scelta degli iper-parametri effettuata per ottenere i risultati di cui sopra.

VGG16	**Precision**	**Recall**	**F1-score**
negative	0.955 +/- 0.026	0.924 +/- 0.029	0.939 +/- 0.016
positive	0.927 +/- 0.025	0.956 +/- 0.027	0.941 +/- 0.015

Tabella 4: Precision, Recall e F1-score per l'architettura VGG16 + strati densi - binario

VGG19	**Precision**	**Recall**	**F1-score**
negative	0.963 +/- 0.015	0.918 +/- 0.042	0.939 +/- 0.023
positive	0.923 +/- 0.035	0.965 +/- 0.015	0.943 +/- 0.019

Tabella 5: Precision, Recall e F1-score per l'architettura VGG19 + strati densi - binario

DenseNet121	**Precision**	**Recall**	**F1-score**
negative	0.938 +/- 0.033	0.914 +/- 0.042	0.925 +/- 0.025
positive	0.918 +/- 0.036	0.938 +/- 0.035	0.927 +/- 0.028

Tabella 6: Precision, Recall e F1-score per l'architettura DenseNet121 + strati densi - binario

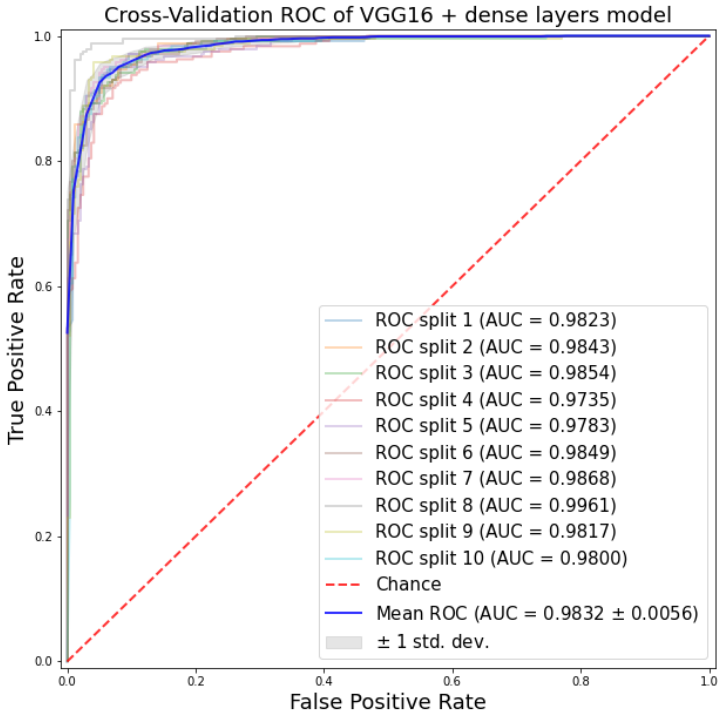

Figura 28: Curva ROC VGG16 + strati densi - binario

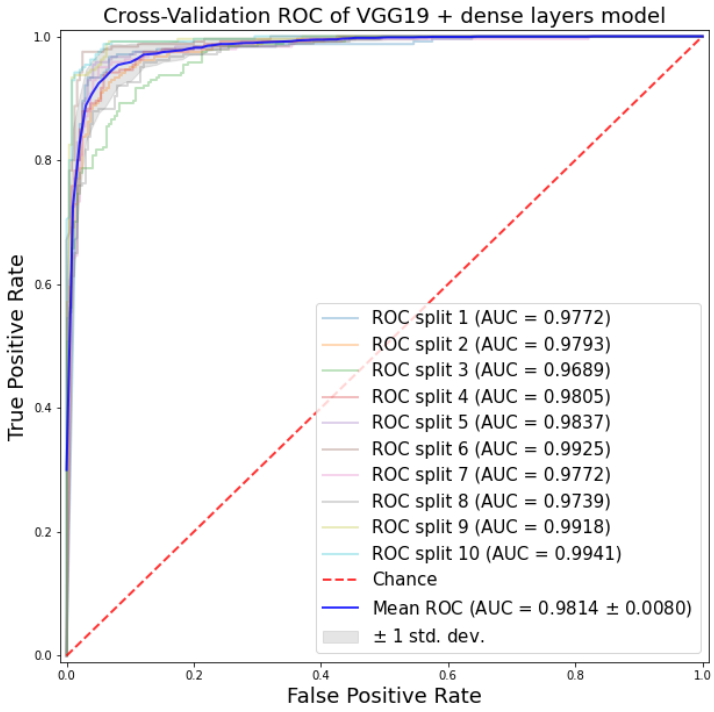

Figura 29: Curva ROC VGG19 + strati densi - binario

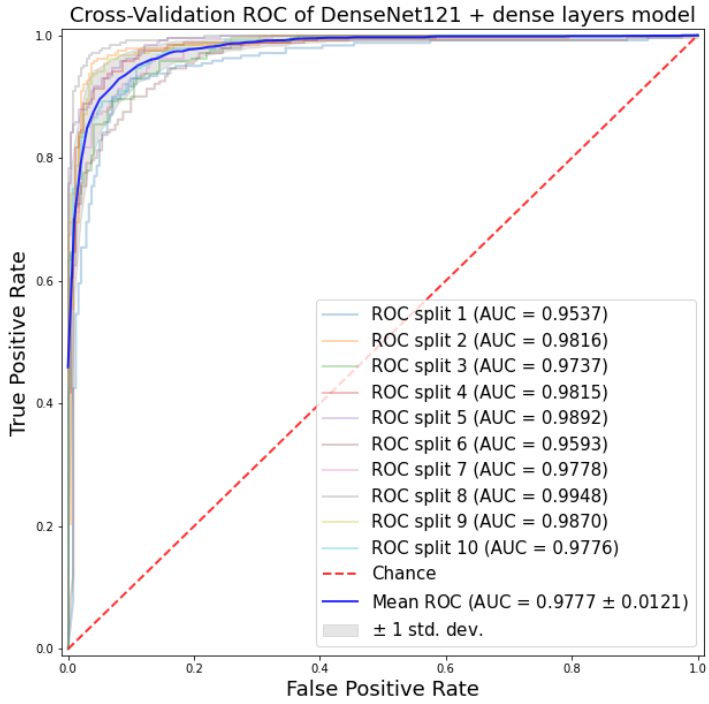

Figura 30: Curva ROC DenseNet121 + strati densi - binario

Per ottenere i risultati precedenti, vale a dire i migliori per le diverse combinazioni tentate sulle stesse tipologie di reti, i tre test sono stati effettuati con:

- ultimi 8 layer scongelati;
- immagini RGB;
- assenza di normalizzazione dell'illuminazione;
- data augmentation (horizontal, vertical flip);
- limite di 100 epoche;
- 10 split;
- batchsize 32;
- random state 0.

Inoltre, come ottimizzatore si è utilizzato l'SGD con learning rate = 0.0001 per la rete VGG16, SGD con lr = 0.001 per la VGG19, mentre Adam con lr = 0.0001 per la DenseNet121.

Nelle seguenti tabelle sono elencati gli strati utilizzati per la classificazione per i tre differenti modelli:

Layer Type	Output Shape	Parameter #
VGG16 (Functional)	(None, 7, 7, 512)	14714688
GlobalAveragePooling2D	(None, 512)	0
Flatten	(None, 512)	0
Dense, 512 units, ReLU activation	(None, 512)	262656
Dense, 2 units, Softmax activation	(None, 2)	1026

Tabella 7: VGG16 + fully connected layers - binario

Layer Type	Output Shape	Parameter #
VGG19 (Functional)	(None, 7, 7, 512)	20024384
GlobalAveragePooling2D	(None, 512)	0
Flatten	(None, 512)	0
Dense, 512 units, ReLU activation	(None, 512)	262656
Dense, 2 units, Softmax activation	(None, 2)	1026

Tabella 8: VGG19 + fully connected layers - binario

Layer Type	Output Shape	Parameter #
DenseNet121 (Functional)	(None, 7, 7, 1024)	7037504
GlobalAveragePooling2D	(None, 1024)	0
Flatten	(None, 1024)	0
Dense, 512 units, ReLU activation	(None, 512)	524800
Dense, 2 units, Softmax activation	(None, 2)	1026

Tabella 9: DenseNet121 + fully connected layers - binario

5.4.2 Dataset a tre classi

Dopo aver constatato le potenzialità di alcune architetture per il problema di classificazione binario, si è deciso di testare le migliori combinazioni anche nel caso a tre classi. Inoltre, sebbene nel problema binario alcuni modelli non abbiano fornito prestazioni valide, per completezza le stesse prove sono state effettuate per ciascuno di essi, anche in questo caso testando differenti combinazioni al fine di trovare le migliori per aumentare i valori dei criteri monitorati.

Come per gli esperimenti condotti in precedenza, tutti i test sono stati svolti con dataset con immagini aumentate, sia per la prima versione del dataset a tre classi che per la seconda, con la differenza che nel primo caso sono stati effettuati sia flip orizzontale che verticale, risultanti in 750 immagini per classe, mentre per il secondo si è adottato il solo flip orizzontale, ottenendo 1000 immagini per classe. Nel seguito verranno prima presentati i risultati e le architetture che hanno fornito le performance migliori per la versione 1, successivamente le stesse modalità saranno impiegate per descrivere i risultati con la versione 2.

Le prestazioni migliori di classificazione a tre classi relative a ciascuna rete, per la versione aumentata del dataset nella sua prima versione, sono riportate in Tabella 10.

Anche in questo caso, per ogni esperimento sono state effettuate più prove con split differenti del dataset e i criteri di valutazione riportati rappresentano il valor medio dei valori raggiunti per i singoli split.

Architettura	Accuratezza	Loss
VGG16	0.966 +/- 0.009	0.097 +/- 0.024
VGG19	0.964 +/- 0.012	0.097 +/- 0.032
ResNet50V2	0.734 +/- 0.021	0.664 +/- 0.061
ResNet101V2	0.712 +/- 0.018	0.697 +/- 0.049
ResNet152V2	0.691 +/- 0.017	0.711 +/- 0.029
InceptionV3	0.679 +/- 0.021	1.041 +/- 0.099
DenseNet121	0.925 +/- 0.013	0.215 +/- 0.036
MobileNetV2	0.861 +/- 0.015	0.374 +/- 0.027
Xception	0.815 +/- 0.029	0.491 +/- 0.043
NASNetMobile	0.576 +/- 0.029	0.927 +/- 0.037

Tabella 10: Confronto delle performance per il task di classificazione a tre classi - 3 classi, versione 1

Come è possibile notare dai risultati in tabella, anche per il task di classificazione a tre classi valgono le stesse considerazioni del caso binario, con i migliori modelli rappresentati dalle VGG16 e VGG19, seguite dalla DenseNet121, ciascuna con un valore di accuratezza superiore al 90%.
Nel seguito si riportano i valori di Precisione, Recall e F1-score ottenuti per le tre classi presenti nel dataset per le tre reti migliori, mentre nelle Figure 31 - 33 sono rappresentate le curve ROC con il rispettivo valore di AUC per i 10 split di ogni modello.

VGG16	**Precision**	**Recall**	**F1-score**
negative	0.954 +/- 0.021	0.977 +/- 0.017	0.965 +/- 0.014
nrd	0.973 +/- 0.016	0.961 +/- 0.026	0.967 +/- 0.014
rd	0.972 +/- 0.011	0.961 +/- 0.016	0.966 +/- 0.007

Tabella 11: Precision, Recall e F1-score per l'architettura VGG16 + strati densi - 3 classi, versione 1

VGG19	**Precision**	**Recall**	**F1-score**
negative	0.947 +/- 0.029	0.974 +/- 0.014	0.961 +/- 0.016
nrd	0.971 +/- 0.011	0.966 +/- 0.028	0.968 +/- 0.013
rd	0.977 +/- 0.011	0.953 +/- 0.023	0.964 +/- 0.011

Tabella 12: Precision, Recall e F1-score per l'architettura VGG19 + strati densi - 3 classi, versione 1

DenseNet121	**Precision**	**Recall**	**F1-score**
negative	0.908 +/- 0.021	0.914 +/- 0.038	0.911 +/- 0.015
nrd	0.947 +/- 0.029	0.935 +/- 0.022	0.941 +/- 0.012
rd	0.925 +/- 0.028	0.926 +/- 0.021	0.925 +/- 0.018

Tabella 13: Precision, Recall e F1-score per l'architettura DenseNet121 + strati densi - 3 classi, versione 1

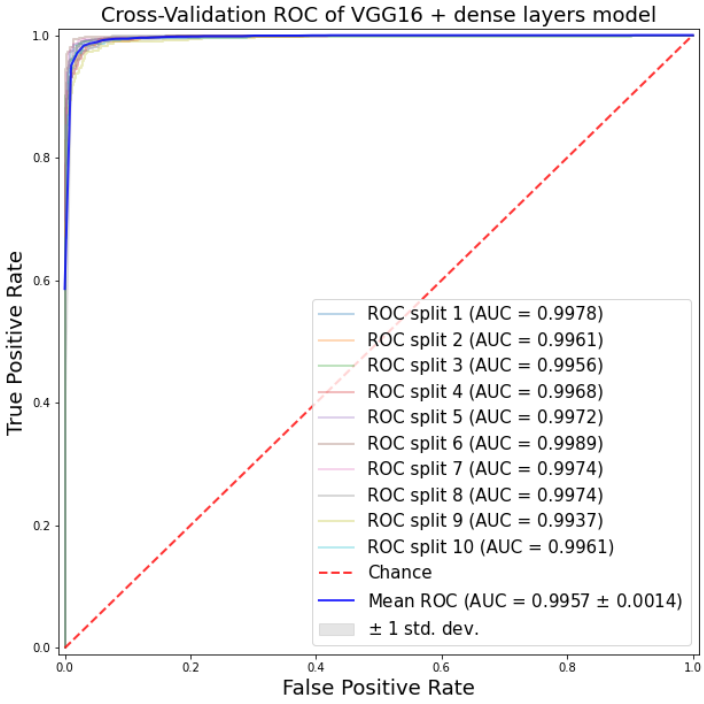

Figura 31: Curva ROC VGG16 + strati densi - 3 classi, versione 1

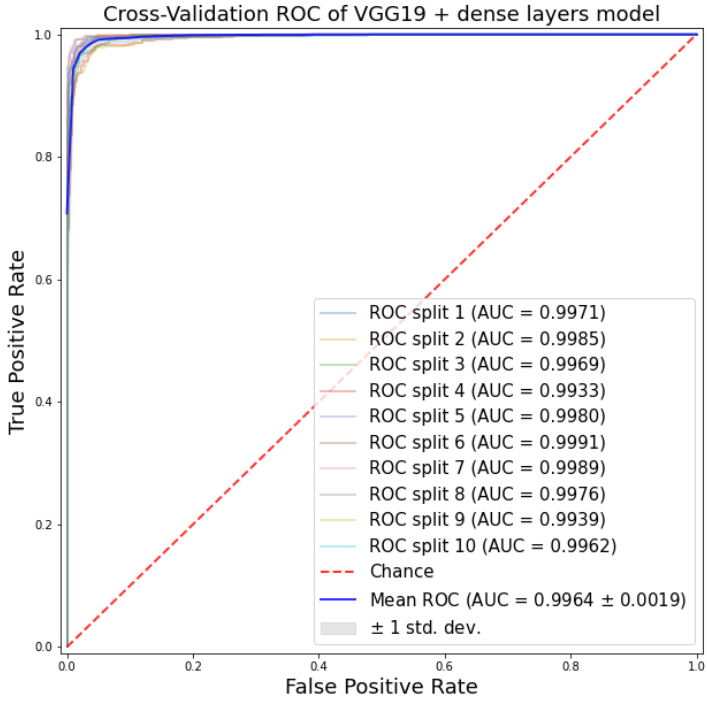

Figura 32: Curva ROC VGG19 + strati densi - 3 classi, versione 1

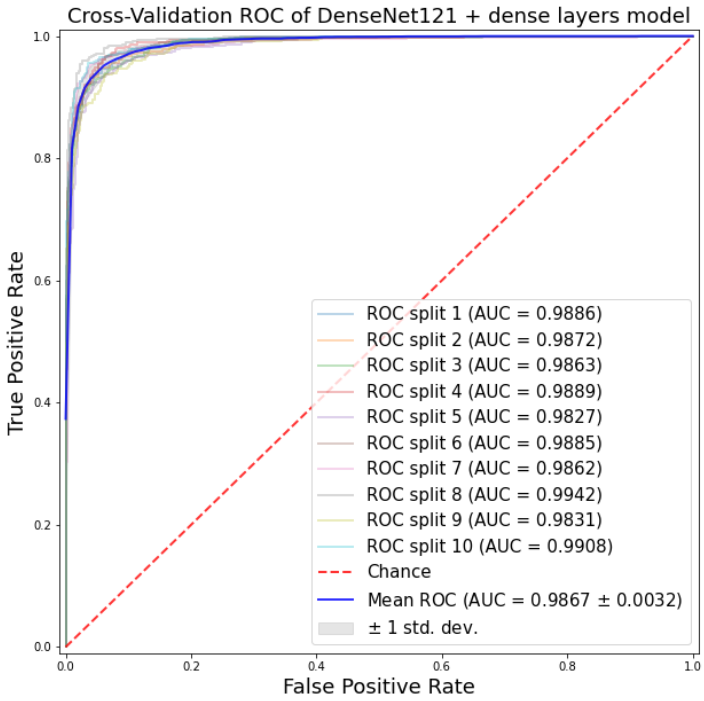

Figura 33: Curva ROC DenseNet121 + strati densi - 3 classi, versione 1

Come accaduto per il caso binario, si riportano le scelte che hanno portato ad ottenere i risultati mostrati:

- ultimi 8 layer scongelati;
- immagini RGB;
- assenza di normalizzazione dell'illuminazione;
- data augmentation (horizontal, vertical flip);
- limite di 100 epoche;
- 10 split;
- batchsize 32;
- random state 0.

Come ottimizzatore si è utilizzato l'SGD con learning rate = 0.001 per la VGG16 e VGG19, mentre Adam con learning rate = 0.0001 per la DenseNet121.

Nelle seguenti tabelle sono elencati gli strati utilizzati per la classificazione per i tre differenti modelli, che rappresentano la scelta migliore sia per la prima versione del dataset a tre classi che per la seconda, i cui risultati saranno presentati nel seguito:

Layer Type	Output Shape	Parameter #
VGG16 (Functional)	(None, 7, 7, 512)	14714688
GlobalAveragePooling2D	(None, 512)	0
BatchNormalization	(None, 512)	2048
Flatten	(None, 512)	0
Dense, 512 units, ReLU activation	(None, 512)	262656
Dense, 3 units, Softmax activation	(None, 3)	1539

Tabella 14: VGG16 + fully connected layers - 3 classi, versione 1/2

Layer Type	Output Shape	Parameter #
VGG19 (Functional)	(None, 7, 7, 512)	20024384
GlobalAveragePooling2D	(None, 512)	0
BatchNormalization	(None, 512)	2048
Flatten	(None, 512)	0
Dense, 512 units, ReLU activation	(None, 512)	262656
Dense, 3 units, Softmax activation	(None, 3)	1539

Tabella 15: VGG19 + fully connected layers - 3 classi, versione 1/2

Layer Type	Output Shape	Parameter #
DenseNet121 (Functional)	(None, 7, 7, 1024)	7037504
GlobalAveragePooling2D	(None, 1024)	0
Flatten	(None, 1024)	0
Dense, 512 units, ReLU activation	(None, 512)	524800
Dense, 3 units, Softmax activation	(None, 3)	1539

Tabella 16: DenseNet121 + fully connected layers - 3 classi, versione 1/2

Infine, si riportano le stesse tipologie di risultati per la terza e ultima versione del dataset, vale a dire costituito da tre classi e un numero maggiore di tipologie di difetti per la classe "non_recoverable_defects", oltre ad avere 500 immagini per classe nella sua versione originale. In quest'ultimo caso, a causa di limitazioni nell'uso di Google Colab, non è stato possibile adottare entrambi i flip orizzontale e verticale. Si è dunque scelto di effettuare solo il primo, consentendo comunque di ottenere un numero di immagini superiore al caso precedente, con 1000 campioni per ciascuna classe. Le scelte che hanno condizionato i risultati finali sono state le medesime della versione 1, pertanto di seguito si riporteranno i soli risultati, senza ripetere le valutazioni a monte.

Architettura	Accuratezza	Loss
VGG16	0.958 +/- 0.009	0.137 +/- 0.027
VGG19	0.951 +/- 0.009	0.149 +/- 0.033
ResNet50V2	0.747 +/- 0.015	0.638 +/- 0.031
ResNet101V2	0.715 +/- 0.018	0.689 +/- 0.028
ResNet152V2	0.735 +/- 0.015	0.651 +/- 0.033
InceptionV3	0.681 +/- 0.031	1.059 +/- 0.096
DenseNet121	0.932 +/- 0.011	0.211 +/- 0.024
MobileNetV2	0.823 +/- 0.016	0.465 +/- 0.037
Xception	0.759 +/- 0.017	0.661 +/- 0.039
NASNetMobile	0.621 +/- 0.018	0.892 +/- 0.033

Tabella 17: Confronto delle performance per il task di classificazione a tre classi - 3 classi, versione 2

VGG16	Precision	Recall	F1-score
negative	0.939 +/- 0.022	0.971 +/- 0.014	0.954 +/- 0.011
nrd	0.968 +/- 0.011	0.947 +/- 0.016	0.957 +/- 0.009
rd	0.969 +/- 0.014	0.957 +/- 0.021	0.963 +/- 0.012

Tabella 18: Precision, Recall e F1-score per l'architettura VGG16 + strati densi - 3 classi, versione 2

VGG19	Precision	Recall	F1-score
negative	0.929 +/- 0.017	0.963 +/- 0.011	0.946 +/- 0.013
nrd	0.963 +/- 0.014	0.938 +/- 0.017	0.951 +/- 0.011
rd	0.962 +/- 0.016	0.952 +/- 0.009	0.957 +/- 0.011

Tabella 19: Precision, Recall e F1-score per l'architettura VGG19 + strati densi - 3 classi, versione 2

DenseNet121	Precision	Recall	F1-score
negative	0.912 +/- 0.026	0.926 +/- 0.022	0.919 +/- 0.014
nrd	0.935 +/- 0.021	0.934 +/- 0.021	0.934 +/- 0.009
rd	0.952 +/- 0.012	0.937 +/- 0.021	0.944 +/- 0.011

Tabella 20: Precision, Recall e F1-score per l'architettura DenseNet121 + strati densi - 3 classi, versione 2

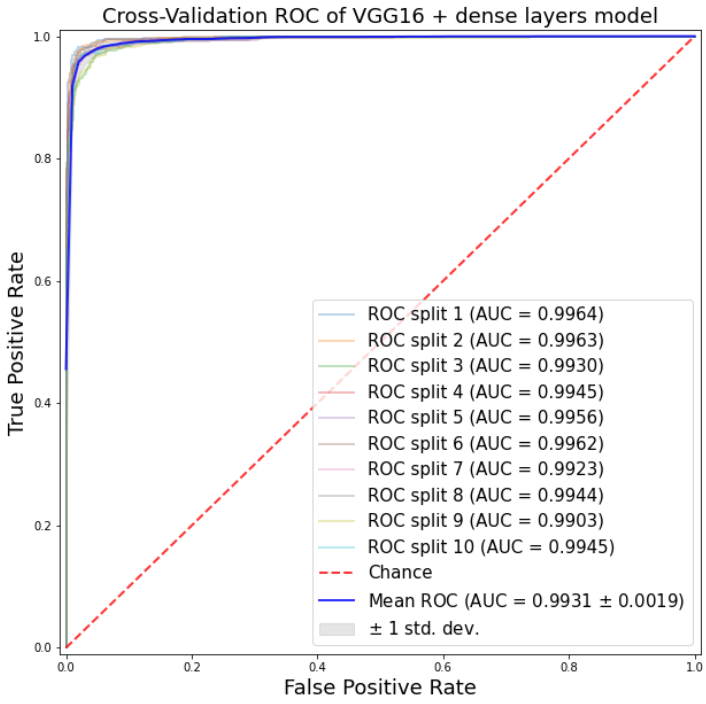

Figura 34: Curva ROC VGG16 + strati densi - 3 classi, versione 2

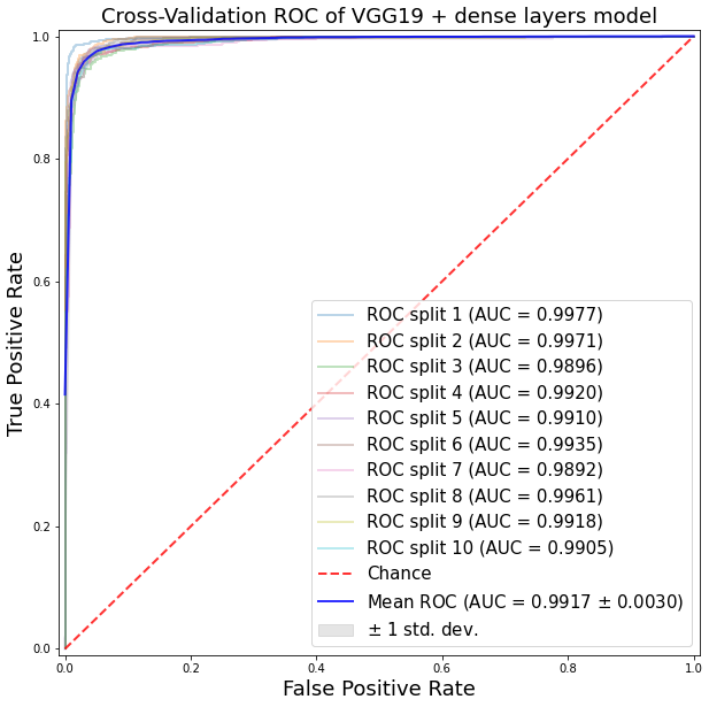

Figura 35: Curva ROC VGG19 + strati densi - 3 classi, versione 2

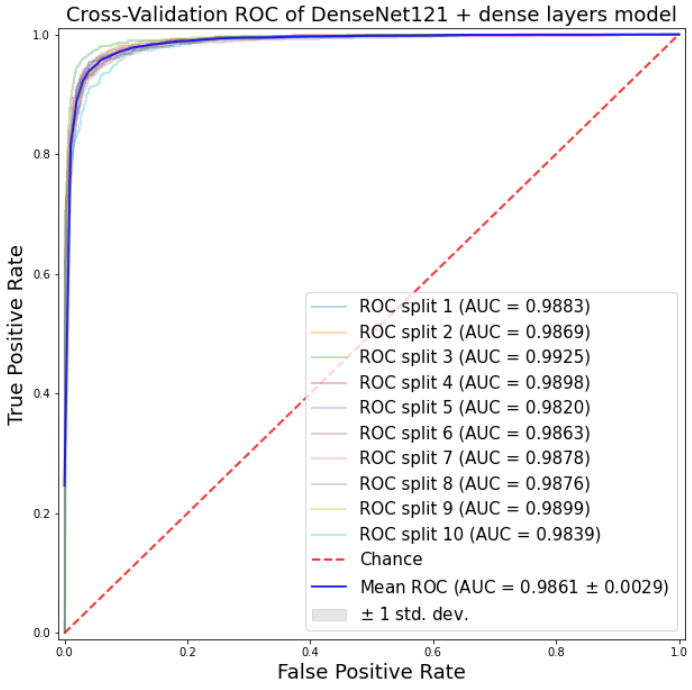

Figura 36: Curva ROC DenseNet121 + strati densi - 3 classi, versione 2

6 Discussioni

Nel presente capitolo verranno discusse le performance dei risultati ottenuti per i differenti dataset testati, presentando alcune delle valutazioni effettuate nel tentativo di migliorare i valori monitorati in base ai criteri scelti. Si valuteranno infine dei possibili sviluppi futuri, che potrebbero fornire utili spunti per migliorare quanto fatto durante il periodo di tirocinio e contribuire alla realizzazione di un sistema automatizzato utilizzabile in ambiente industriale.

6.1 Valutazione delle performance

I risultati ottenuti da alcuni dei modelli testati, pre-addestrati con dataset ImageNet e successivamente perfezionati tramite fine-tuning con i dataset proposti, hanno dimostrato come sia possibile affrontare il problema del riconoscimento di difetti superficiali tramite sistemi automatizzati anche nel campo dei materiali compositi. Come riportato nel precedente capitolo, i risultati mostrati sono conseguenza di diverse valutazioni svolte a monte e rappresentano le combinazioni migliori ottenute variando i parametri regolabili e apportando modifiche strutturali ai modelli. Alcune di queste modifiche hanno influito marginalmente, altre hanno avuto un ruolo chiave nel raggiungimento dei valori riportati, in particolare per i modelli VGG16 e VGG19, che hanno fornito le prestazioni migliori per ciascun dataset utilizzato in ingresso.

Entrando nello specifico e facendo riferimento a quanto descritto nel paragrafo 5.3, la differente scelta di batch size e random state non è risultata determinante, non comportando modificazioni apprezzabili. Ciò rappresenta un elemento positivo, in quanto sta ad indicare che alcuni classificatori sono sufficientemente indipendenti dal particolare split di dati scelto. Per la maggior parte dei test si è scelto come valore di batch size 32 e random state pari a zero. Il numero di epoche e di split in alcuni casi si sono dimostrati più influenti. Ad esempio, per il modello DenseNet121, risultato il terzo più performante sia nel caso binario che multi classe, l'utilizzo di un numero di epoche massimo pari a 50, come impostato per i primi test effettuati, è risultato limitante, in quanto l'impiego di un numero massimo pari a 100 ha permesso di addestrare il modello più a lungo, a differenza della maggior parte delle altre architetture per cui sarebbero state sufficienti circa 20 epoche prima di andare incontro ad overfit. Anche il numero di split, inizialmente impostato pari a 5 e in seguito fissato a 10, ha permesso di scartare alcune configurazioni che nel primo caso davano risultati paragonabili tra i diversi split, mentre nel secondo, avendo a disposizione un maggior numero di

combinazioni di immagini per determinare i set di train, validation e test, mostravano delle sostanziali differenze prestazionali tra split differenti utilizzati per lo stesso modello. L'utilizzo di immagini RGB o in scala di grigi non sono stati elementi rilevanti, avendo ottenuto nei test effettuati risultati paragonabili, mantenendo gli altri parametri invariati. Lo stesso non vale per la normalizzazione dell'illuminazione, inizialmente sempre inserita in quanto ampiamente utilizzata in letteratura in lavori simili; infatti, per ciascun modello, compresi i casi meno performanti, sono stati raggiunti i più alti valori di accuratezza evitando di applicare la normalizzazione, a volte con un incremento del 5% rispetto al medesimo caso con filtro CLAHE applicato al termine della data augmentation. I cambiamenti più significativi e di validità generale sono stati ottenuti introducendo uno strato di Global Average Pooling antecedente agli strati fully connected. Tra i vari test si è provato anche a sostituirlo con un Max Pooling ed inserendo alcuni strati di Dropout, non ottenendo tuttavia risultati apprezzabili. Inoltre, per i modelli VGG16 e VGG19 che hanno fornito le prestazioni migliori per il problema multi classe, l'introduzione di una Batch Normalization successiva allo strato di pooling menzionato ha costituito un ulteriore elemento di miglioramento, consentendo inoltre di ottenere risultati più uniformi in termini di accuratezza, precisione, recall e f1-score tra i diversi split e le diverse classi valutate singolarmente.

Come più volte riportato, i modelli VGG16 e VGG19 hanno fornito risultati molto promettenti e di gran lunga migliori rispetto a modelli più complessi e costituiti da molti più strati quali le ResNet, risultate nel contesto in esame tra le architetture meno performanti. Nei tre casi proposti i due modelli sono stati in grado di raggiungere livelli di accuratezza di circa il 95%, mostrando risultati migliori nel caso a tre classi rispetto al binario, con valori prossimi al 97% per la prima versione testata. Ciò potrebbe risultare sorprendente, in quanto un aumento del numero di classi solitamente comporta un abbassamento delle prestazioni del classificatore. Una possibile spiegazione potrebbe riguardare il maggior numero di immagini utilizzate per l'addestramento, ulteriormente aumentate tramite la data augmentation, e una maggiore separazione tra le classi di difetto, unita ad una migliore qualità delle immagini acquisite. In aggiunta all'accuratezza, i criteri di precisione, recall e f1-score per le diverse classi sono stati caratterizzati da valori molto stabili e uniformi, a dimostrazione della capacità delle due architetture di classificare in maniera robusta le differenti tipologie di difetto. Le performance dei modelli appartenenti alla famiglia ResNet, Inception e NASNet hanno invece riportato risultati deludenti, con NASNet risultata la peggiore in termini di accuratezza per ciascun dataset.

6.2 Sviluppi futuri

Da questo primo tentativo di automatizzare il controllo superficiale di materiali compositi è stato possibile trarre conclusioni positive, avendo dimostrato la bontà di alcune architetture sfruttando la nota tecnica del transfer learning, con modelli per l'estrazione di feature ampiamente utilizzati in letteratura e facili da implementare. Tra i possibili sviluppi futuri, per estendere quanto descritto nella presente trattazione, possono essere incluse le seguenti opzioni:

1. Estendere il numero di classi e migliorare la qualità delle immagini.
2. Utilizzare un robot collaborativo per l'analisi di superfici complesse.
3. Implementare algoritmi per la detection dei difetti.

Con una maggiore disponibilità di componenti che presentino le differenti tipologie di difetti, potrebbe risultare possibile la costruzione di un dataset con una classe per difetto, senza doverne realizzare una versione fortemente sbilanciata. Inoltre, si potrebbero acquisire le foto da cui ricavare le immagini in maniera più standardizzata, facendo uso di telecamere industriali in sostituzione dello smartphone. A tal proposito, quest'ultima applicazione è in fase di valutazione per la costruzione di un quarto dataset, la cui realizzazione potrebbe essere resa possibile grazie alla collaborazione con l'azienda Vision Device di Torrevecchia Teatina (CH), specializzata nella progettazione e realizzazione di sistemi di visione artificiale. La differenza sostanziale rispetto ai casi presentati in precedenza consisterebbe nella metodologia impiegata nell'acquisire le foto, non più tramite smartphone ma per mezzo di un sistema portatile fornito dall'azienda, che può essere spostato sopra le superfici da analizzare. Si tratta di una semplice struttura che monta una telecamera matriciale e un sistema di illuminazione fisso, che consente di ottenere immagini standardizzate, mantenendo le stesse condizioni di luce e distanza tra telecamera e componente.
Un ulteriore possibile sviluppo potrebbe riguardare l'utilizzo di un robot collaborativo per l'analisi delle superfici con curvature significative. Il robot in questione rappresenta la soluzione ideale per poter analizzare molteplici superfici se in possesso della geometria dei componenti, essendo in grado di mantenere l'end effector ad una distanza costante anche in presenza di curve e avendo a disposizione telecamere e sistemi di illuminazione adattabili in base al contesto.
Nel lavoro presentato si è affrontato il task come un problema di classificazione. Per un possibile sviluppo in merito, si potrebbe prendere in considerazione l'implementare algoritmi di object detection per andare a individuare e evidenziare differenti difetti presenti nella stessa immagine,

ottenendo un sistema con delle funzionalità che siano paragonabili ai più moderni e utilizzati sistemi di visione artificiale in ambiente industriale.

7 Conclusioni

In questo studio si è introdotto il problema della rilevazione automatica di difetti superficiali di natura estetica in materiali compositi. Data la novità rappresentata dal tema proposto, sono state effettuate ricerche in letteratura con lo scopo di valutare le tecniche maggiormente utilizzate per l'ispezione di materiali che presentassero una vicinanza tematica. Il metodo proposto per il raggiungimento dell'obiettivo ha riguardato la realizzazione di tre differenti dataset a partire da immagini acquisite su componenti forniti dall'azienda ospitante, il primo di tipo binario e i restanti due a tre classi. Dalla ricerca, è emerso come l'utilizzo di CNN, sfruttando la tecnica del transfer learning, sia tra le soluzioni più adottate e di successo per compiti di classificazione di immagini. La parte sperimentale del lavoro svolto è costituita dalle prestazioni ottenute confrontando dieci differenti architetture di CNN: VGG16, VGG19, ResNet50, ResNet101, ResNet152, InceptionV3, DenseNet121, MobileNet, Xception e NASNetMobile. Le architetture VGG16 e VGG19 hanno prodotto i risultati migliori, con una accuratezza media nella classificazione del 95% per il caso binario e superiore al 96% per il problema multi classe. I risultati ottenuti dimostrano come sia possibile automatizzare il controllo della qualità superficiale nei materiali compositi.

Bibliografia

[1] https://www.hpcomposites.it/

[2] https://www.hpcomposites.it/mercati.html

[3] https://www.hpcomposites.it/azienda.html

[4] https://www.infobuild.it/approfondimenti/materiali-compositi/

[5] https://www.hpcomposites.it/tecnologie.html

[6] Giovanni Serventi, Studio delle cause dell'insorgenza di difetti estetici ed errori dimensionali nella realizzazione di manufatti in fibra di carbonio. Laurea triennale, Università degli studi di Padova, Dipartimento di Ingegneria Industriale, 2011.

[7] https://www.bmw.com/en/performance/carbon-fiber-in-a-car.html

[8] Ajay Kumar "Computer-Vision-Based Fabric Defect Detection: A Survey.", IEEE Transactions on Industrial Electronics, vol. 55, no. 1, January 2008.

[9] C.-C. Hung and I.-C. Chen "Neural-fuzzy classification for fabric defects." Text. Res. J., vol. 71, no. 3, pp, 220-224, 2001.

[10] B.G. Batchelor "Lighting and viewing techniques." in Automated Visual Inspection, B.G. Batchelor, D.A. Hill and D.C. Hodgson, Eds. Amsterdam, The Netherlands: North Holland, 1985.

[11] G. Pang, R. Wong, H. Liu, T. Kwan, A. Kumar "CAVIS: A low-cost fabric defect inspection machine based on Machine Vision.", in Proc. Asian Textile Conf., Aug. 2001, pp. 11-16.

[12] Czimmermann, T., Ciuti, G., Milazzo, M., Chiurazzi, M., Roccella, S., Oddo, C.M., & Dario, P. "Visual-based defect detection and classification approaches for industrial applications—a survey.", Sensors, 20(5), 2020, 1459.

[13] Xie, X. "A review of recent advances in surface defect detection using texture analysis techniques.", Electron. Lett. Comput. Vis. Image Anal. 2008, 7, 1–22.

[14] Ngan, H.Y., Pang, G.K., Yung, N.H. "Automated fabric defect detection–a review.", Image Vis. Comput. 2011, 29, 442–458.

[15] Zhang, Lei, Junfeng Jing and Hongwei Zhang "Fabric defect classification based on LBP and GLCM.", Journal of Fiber Bioengineering and Informatics, 8.1 (2015): 81-89.

[16] Germany, D. Tilda Textile Texture Database. 1996. Available online: https://lmb.informatik.uni-freiburg.de/resources/datasets/tilda.en.html

[17] Sindagi, Vishwanath A. and Sumit Srivastava "Oled panel defect detection using local inlier-outlier ratios and modified LBP.", 2015 14th IAPR International Conference on Machine Vision Applications (MVA). IEEE, 2015.

[18] Chen, Jiahan and Anil K. Jain "A structural approach to identify defects in textured images.", Proceedings of the 1988 IEEE International Conference on Systems, Man, and Cybernetics. Vol. 1. IEEE, 1988.

[19] Bennamoun, Mohammed and Adriana Bodnarova "Automatic visual inspection and flaw detection in textile materials: Past, present and future.", SMC'98 Conference Proceedings. 1998 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No. 98CH36218). Vol. 5. IEEE, 1998.

[20] Neubauer, C. "Segmentation of defects in textile fabric.", In Proceedings of the 11th IAPR International Conference on Pattern Recognition, The Hague, The Netherlands, 30 August–3 September 1992; pp. 688–691.

[21] Gai, S. "New banknote defect detection algorithm using quaternion wavelet transform.", Neurocomputing 2016, 196, 133–139.

[22] Kumar, A., Pang, G.K. "Fabric defect segmentation using multichannel blob detectors.", Opt. Eng. 2000, 39, 3176–3191.

[23] Yang, X., Pang, G., Yung, N. "Discriminative fabric defect detection using adaptive wavelets.", Opt. Eng. 2002, 41, 3116–3125.

[24] Hu, B., Wang, J., Huang, X. "Fabric defect detection based on multiple fractal features and support vector data description.", Eng. Appl. Artif. Intell. 2009, 22, 224–235.

[25] Hajimowlana, S.H., Muscedere, R., Jullien, G.A., Roberts, J.W. "1D autoregressive modeling for defect detection in web inspection systems.", In Proceedings of the 1998 Midwest Symposium on Circuits and Systems (Cat. No. 98CB36268), Notre Dame, IN, USA, 9–12 August 1998; pp. 318–321

[26] Xi, J., Shentu, L., Hu, J., Li, M. "Automated surface inspection for steel products using computer vision approach.", Appl. Opt. 2017, 56, 184–192.

[27] Zhao, Z.Q., Zheng, P., Xu, S.T., Wu, X. "Object detection with deep learning: A review.", IEEE Trans. Neural Netw. Learn. Syst. 2019, 30, 3212–3232.

[28] Yang, Jing, et al. "Using deep learning to detect defects in manufacturing: a comprehensive survey and current challenges.", Materials 13.24 (2020): 5755.

[29] Brackenbury, D., Brilakis, I., DeJong, M. "Automated Defect Detection For Masonry Arch Bridges.", International Conference on Smart Infrastructure and Cons.

[30] An, Meng, et al. "Fabric defect detection using deep learning: An Improved Faster R-approach.", 2020, International Conference on Computer Vision, Image and Deep Learning (CVIDL), IEEE, 2020.
[31] Zhu, Zongwei, et al. "Modified densenet for automatic fabric defect detection with edge computing for minimizing latency.", IEEE Internet of Things Journal 7.10 (2020): 9623-9636.
[32] Liu, Juhua, et al. "Multistage GAN for fabric defect detection.", IEEE Transactions on Image Processing, 29 (2019): 3388-3400.
[33] Li, Yundong, Weigang Zhao and Jiahao Pan. "Deformable patterned fabric defect detection with fisher criterion-based deep learning.", IEEE Transactions on Automation Science and Engineering, 14.2 (2016): 1256-1264.
[34] Urbonas, Augustas, et al. "Automated identification of wood veneer surface defects using faster region-based convolutional neural network with data augmentation and transfer learning.", Applied Sciences, 9.22 (2019): 4898.
[35] Gao, Yiping, et al. "A multilevel information fusion-based deep learning method for vision-based defect recognition.", IEEE Transactions on Instrumentation and Measurement, 69.7 (2019): 3980-3991.
[36] Chen, Wen, et al. "A new ensemble approach based on deep convolutional neural networks for steel surface defect classification.", Procedia CIRP, 72 (2018): 1069-1072.
[37] Cha, Y.J., Choi, W., Suh, G., Mahmoudkhani, S., Büyüköztürk, O. "Autonomous structural visual inspection using region-based deep learning for detecting multiple damage types.", Comput.-Aided Civ. Infrastruct. Eng. 2018, 33, 731–747.
[38] Wang, Y., Liu, M., Zheng, P., Yang, H., Zou, J. "A smart surface inspection system using faster R-CNN in cloud-edge computing environment.", Adv. Eng. Inf. 2020, 43, 101037.
[39] Liu, Y.T., Yang, Y.N., Chao, W., Xu, X.Y., Zhang, T. "Research on Surface Defect Detection Based on Semantic Segmentation.", DEStech Trans. Comput. Sci. Eng. 2019.
[40] Yang, Y., Pan, L., Ma, J., Yang, R., Zhu, Y., Yang, Y., Zhang, L. "A High-Performance Deep Learning Algorithm for the Automated Optical Inspection of Laser Welding.", Appl. Sci. 2020, 10, 933.
[41] Bang, Hyun-Tae, Solmoi Park, and Haemin Jeon. "Defect identification in composite materials via thermography and deep learning techniques." Composite Structures 246 (2020): 112405.
[42] Saeed, Numan, et al. "Automatic defects detection in CFRP thermograms, using convolutional neural networks and transfer learning." Infrared Physics & Technology 102 (2019): 103048.

[43] Marani, R., et al. "Modeling and classification of defects in CFRP laminates by thermal non-destructive testing." Composites Part B: Engineering 135 (2018): 129-141.

[44] Marani, Roberto, et al. "Deep learning for defect characterization in composite laminates inspected by step-heating thermography." Optics and Lasers in Engineering 145 (2021): 106679.

[45] Zheng, Kaiyi, Yu-Sung Chang, and Yuan Yao. "Defect detection in CFRP structures using pulsed thermographic data enhanced by penalized least squares methods." Composites Part B: Engineering 79 (2015): 351-358.

[46] Liu, Kaixin, et al. "Enhanced Defect Detection in Carbon Fiber Reinforced Polymer Composites via Generative Kernel Principal Component Thermography." Polymers 13.5 (2021): 825.

[47] Margraf, Andreas, et al. "An evolutionary learning approach to self-configuring image pipelines in the context of carbon fiber fault detection." 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA). IEEE, 2017.

[48] Gong, Yanfeng, et al. "A deep transfer learning model for inclusion defect detection of aeronautics composite materials." Composite Structures 252 (2020): 112681.

[49] https://www.visiondevice.com/it/

[50] LeCun, Yann, et al. "Object recognition with gradient-based learning." Shape, contour and grouping in computer vision. Springer, Berlin, Heidelberg, 1999. 319-345.

[51] LeCun, Yann, et al. "Backpropagation applied to handwritten zip code recognition." Neural computation, 1.4 (1989): 541-551.

[52] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems, 25 (2012): 1097-1105.

[53] Rudakov, Nikolay, et al. "Detection of mechanical damages in sawn timber using convolutional neural networks." German Conference on Pattern Recognition. Springer, Cham, 2018.

[54] Albelwi, Saleh, and Ausif Mahmood. "A framework for designing the architectures of deep convolutional neural networks." Entropy, 19.6 (2017): 242.

[55] Pan, Sinno Jialin, and Qiang Yang. "A survey on transfer learning." IEEE Transactions on knowledge and data engineering, 22.10 (2009): 1345-1359.

[56] Hinton, Geoffrey E., et al. "Improving neural networks by preventing co-adaptation of feature detectors." arXiv preprint arXiv:1207.0580, (2012).

[57] https://www.robots.ox.ac.uk/~vgg/

[58] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).

[59] https://towardsdatascience.com/step-by-step-vgg16-implementation-in-keras-for-beginners-a833c686ae6c

[60] Zheng, Yufeng, Clifford Yang, and Alex Merkulov. "Breast cancer screening using convolutional neural network and follow-up digital mammography." Computational Imaging III. Vol. 10669. International Society for Optics and Photonics, 2018.

[61] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[62] https://www.image-net.org/challenges/LSVRC/2015/index.php

[63] Szegedy, Christian, et al. "Going deeper with convolutions." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

[64] Szegedy, Christian, et al. "Rethinking the inception architecture for computer vision." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[65] Huang, Gao, et al. "Densely connected convolutional networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

[66] Howard, Andrew G., et al. "Mobilenets: Efficient convolutional neural networks for mobile vision applications." arXiv preprint arXiv:1704.04861 (2017).

[67] Chollet, François. "Xception: Deep learning with depthwise separable convolutions." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

[68] Zoph, Barret, et al. "Learning transferable architectures for scalable image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.

[69] https://colab.research.google.com/

[70] https://www.python.org/

[71] https://www.tensorflow.org/

[72] https://keras.io/

[73] https://scikit-learn.org/stable/

[74] https://numpy.org/

[75] https://matplotlib.org/

[76] https://scipy.org/

[77] https://opencv.org/

[78] Zuiderveld, Karel. "Contrast limited adaptive histogram equalization." Graphics gems (1994): 474-485.

[79] https://www.image-net.org/