



Università degli Studi di Salerno

Dipartimento di Informatica

Corso di Laurea Triennale in Informatica

Analisi delle immagini attraverso Reti Neurali Convoluzionali per l'Individuazione di *Fake News*

Relatori

Prof. Andrea Francesco Abate

Prof. Fabio Narducci

Candidata

Mary Cerullo

Matricola: 0512105133

Indice

Abstract	1
1 Introduzione	2
1.1 Obiettivi	2
1.2 Il problema delle Fake News	4
1.3 Campi applicativi	5
2 Stato dell'arte	7
2.1 Image Forgery Detection	7
2.1.1 Tecniche per la manipolazione delle immagini	8
2.1.2 Reti Neurali Convoluzionali e Transfer Learning	10
2.1.3 Risultati ottenuti	11
3 Approccio tecnico	16
3.1 Architettura della CNN proposta	16
3.1.1 Il problema dell'overfitting	17
3.1.2 Training	18
3.1.3 Impatto sulle prestazioni: SGD vs Adam Optimizer	21
4 Esperimenti	25
4.1 Analisi dei dataset utilizzati: CASIA v2.0 vs NC16	25
4.2 Sperimentazione: motivazioni	26
4.3 Pipeline del sistema	27
4.4 Fase 1: esperimenti basati sull'utilizzo di SGD	28
4.4.1 Esperimento 1	28
4.4.2 Esperimento 2	30
4.4.3 Esperimento 3	32
4.5 Fase 2: esperimenti basati sull'utilizzo di Adam	33
4.5.1 Esperimento 1	33
4.5.2 Esperimento 2	34
4.5.3 Esperimento 3	35
4.6 Considerazioni e risultati finali	37
5 Conclusioni	38

Indice delle figure

1.1 Esempio di contraffazione storica, sulla destra l'immagine originale ritraente Stalin e Nikola Yezhov, sulla sinistra l'immagine dopo le modifiche	3
1.2 Esempio di contraffazione storica. Sulla destra l'immagine originale, sulla sinistra Mussolini senza l'aiuto dell'uomo impegnato a tenere fermo il cavallo	3
1.3 Esempio di contraffazione storica - Tydings con il capo del Partito Comunista Americano	4
2.1 Esempio di <i>digital watermarking</i>	7
2.2 Tecniche per l'alterazione di immagini	8
2.3 Esempio applicazione tecnica <i>copy-move</i>	9
2.4 Esempio applicazione tecnica <i>splicing</i>	9
2.5 Esempio applicazione tecnica <i>removal</i>	10
2.6 Architettura di una CNN	11
2.7 Error Level Analysis	12
2.8 Architettura VGG16	14
3.1 Architettura della CNN utilizzata	17
3.2 Esempio grafico del fenomeno dell' <i>overfitting</i>	18
3.3 Estrazione delle <i>patch</i>	19
3.4 Algoritmo SVM	20
3.5 SGD: <i>learning rate</i>	22
3.6 SGD: <i>momentum</i>	23
3.7 Accuratezza: confronto tra i due ottimizzatori Adam ed SGD	24
4.1 Campione modificato appartenente al dataset NC16	26
4.2 Campioni appartenenti al dataset CASIA V2.0	26
4.3 Esempio di organizzazione di una <i>confusion matrix</i>	27
4.4 Pipeline del sistema	27
4.5 <i>training loss</i> del primo esperimento calcolato per i due <i>dataset</i>	29
4.6 <i>training loss</i> del secondo esperimento calcolato per i due <i>dataset</i>	32
4.7 Esempio immagini classificate erroneamente dal modello.	37

Indice delle tabelle

2.1 Risultati ottenuti in letteratura sul <i>dataset</i> CASIA V2.0 per il problema dell' <i>Image Forgery</i>	15
4.1 Esperimento 1: la tabella contiene i parametri utilizzati durante la fase di training	28
4.2 Esperimento 1: la tabella contiene i risultati ottenuti a seguito del training e della classificazione	28
4.3 Esperimento 1: <i>confusion matrix</i> contenente i campioni correttamente classificati e quelli classificati in maniera erronea per il <i>dataset</i> CASIA V2.0	28
4.4 Esperimento 1: <i>confusion matrix</i> contenente i campioni correttamente classificati e quelli classificati in maniera erronea per il <i>dataset</i> NC16	29
4.5 Esperimento 2: la tabella contiene i parametri utilizzati durante la fase di training	30
4.6 Esperimento 2: la tabella contiene i risultati ottenuti a seguito del training e della classificazione	30
4.7 Esperimento 2: <i>confusion matrix</i> contenente i campioni correttamente classificati e quelli classificati in maniera erronea per il <i>dataset</i> CASIA V2.0	31
4.8 Esperimento 2: <i>confusion matrix</i> contenente i campioni correttamente classificati e quelli classificati in maniera erronea per il <i>dataset</i> NC16	31
4.9 Esperimento 3: accuratezza del modello in termini di generalizzazione	33
4.10 Esperimento 1: la tabella contiene i parametri utilizzati durante la fase di training	33
4.11 Esperimento 1: la tabella contiene i risultati ottenuti a seguito del training e della classificazione	34
4.12 Esperimento 1: <i>confusion matrix</i> contenente i campioni correttamente classificati e quelli classificati in maniera erronea per il <i>dataset</i> CASIA V2.0	34
4.13 Esperimento 1: <i>confusion matrix</i> contenente i campioni correttamente classificati e quelli classificati in maniera erronea per il <i>dataset</i> NC16	34
4.14 Esperimento 2: la tabella contiene i parametri utilizzati durante la fase di training	35
4.15 Esperimento 2: la tabella contiene i risultati ottenuti a seguito del training e della classificazione	35
4.16 Esperimento 2: <i>confusion matrix</i> contenente i campioni correttamente classificati e quelli classificati in maniera erronea per il <i>dataset</i> NC16	35
4.17 Esperimento 3: generalizzazione - confronto risultati tra i due ottimizzatori Adam ed SGD	36

4.18 Risultati, in termini di accuratezza della classificazione, degli esperimenti effettuati con il <i>dataset</i> CASIA V2.0 completo e gli esperimenti, fulcro dell'elaborato e del lavoro di tesi proposto, utilizzando il <i>dataset</i> CASIA 2.0 incompleto	36
5.1 La tabella mostra la sintesi di tutti i risultati ottenuti durante la sperimentazione sia tenendo conto del valore del <i>learning rate</i> utilizzato, che considerando i differenti ottimizzatori impiegati	39

Abstract

Il problema delle *fake news* è un problema largamente diffuso. Negli ultimi anni, l'utilizzo dei social ne ha incrementato la circolazione, diventando, di fatto, una problematica sociale che mina costantemente il diritto ad una corretta informazione. Tra le metodologie diffuse per favorirne la diffusione vi è l'utilizzo di immagini falsificate e volutamente fuorvianti. A questo scopo, da molti anni, sono disponibili una serie di tecnologie atte ad arginare questo problema. Prima tra tutte, l'utilizzo dell'intelligenza artificiale che, tramite l'analisi delle immagini e l'utilizzo di modelli capaci di classificare le stesse sulla base delle alterazioni subite, costituisce un importante mezzo di difesa. L'obiettivo dello studio è proprio quello di determinare che risultati si possano ottenere considerando due *dataset* differenti, sia in termini numerici che per difficoltà di riconoscimento delle alterazioni delle immagini contenute al loro interno, utilizzando una Rete Neurale Convoluzionale. L'approccio utilizzato è di tipo sperimentale e suddiviso in due fasi differenti. La prima, utilizza un ottimizzatore, e la seconda, valutandone eventuali migliorie o peggioramenti, utilizza un ottimizzatore differente. I risultati ottenuti mostrano come, non solo la scelta di impiegare un numero minore di campioni per il *dataset* avente una quantità maggiore di immagini al suo interno (CASIA V2.0) abbia fortemente influenzato il modello, ma che per il *dataset* più impegnativo (NC16) si siano ottenuti risultati migliori, in termini di classificazione finale, diversamente da quanto si potesse pensare. Tuttavia, la spiegazione di tale fenomeno è riconducibile proprio all'utilizzo parziale, e non totale, del numero di campioni presenti nel *dataset* CASIA V2.0. Un'ulteriore conferma la si può trovare nel fatto che la scelta di utilizzare due ottimizzatori differenti si è mostrata sì utile per il *dataset* NC16, mentre si è mostrata decisamente meno efficace nel secondo caso.

Capitolo 1

Introduzione

1.1 Obiettivi

Lo scopo del lavoro di tesi effettuato è quello di fornire una panoramica sul problema delle *fake news* e sulle tecnologie presenti in letteratura per contrastare questo fenomeno. In particolare, su come è possibile utilizzare le immagini e il loro contenuto, tramite Reti Neurali Convoluzionali, per poterne effettuare una classificazione attraverso eventuali alterazioni subite. Le motivazioni dietro questa scelta derivano dall'estrema delicatezza del periodo storico che stiamo vivendo e dall'estrema facilità con cui è possibile manipolare le immagini e, di conseguenza, l'informazione. Sebbene possiamo considerare moderna l'adozione di metodologie avanzate per individuare le alterazioni nelle immagini, i crimini legati alla contraffazione sono tutt'altro che recenti. Per meglio comprendere il motivo e l'importanza dello studio proposto, di seguito vengono riportate una serie di contraffazioni storiche che hanno avuto un impatto rilevante sull'opinione pubblica. Basti pensare a come le fotografie divennero una vera e propria arma nelle mani di Stalin, durante il periodo delle Grandi Purghe e, sebbene non ci fossero i mezzi disponibili oggi, riuscì comunque ad eliminare tracce dei suoi nemici dalle fotografie scattatogli. In figura 1.1, come si può notare, Stalin, tramite alcuni fotoritoccatori dell'epoca, fece eliminare dalla fotografia Nikola Yezhov, un funzionario di polizia che lavorò per lui durante il periodo delle Grandi Purghe. Lo stesso accusò ingiustamente e fece ordinare l'esecuzione di migliaia di funzionari del Partito Comunista e, nel 1938, a seguito della sfiducia di Stalin, dopo essere stato usurpato da uno dei suoi vice, fu denunciato e processato in un tribunale segreto per poi essere giustiziato. [1] E, ancora, lo stesso Mussolini fece rimuovere la persona impegnata a tenergli fermo il cavallo per apparire più eroico, come è possibile notare in figura 1.2. Infine, l'immagine falsificata e rappresentata in figura 1.3, raffigurante il Senatore Tydings in un dialogo con il capo del Partito Comunista Americano, avrebbe addirittura messo in discussione la campagna per la propria rielezione. [2] A tale proposito, la metodologia adottata nella fase sperimentale del lavoro di tesi si è focalizzata sull'utilizzo di Reti Neurali Convoluzionali dove, sulla base di un modello esistente, sono state effettuate una serie di sperimentazioni atte a verificare il comportamento dello stesso sulla problematica delle immagini contraffatte e successivamente come, applicando un approccio diverso, se ne potessero migliorare le prestazioni o meno. In particolare, lo scopo dello studio è quello



Figura 1.1: Esempio di contraffazione storica, sulla destra l'immagine originale ritraente Stalin e Nikola Yezhov, sulla sinistra l'immagine dopo le modifiche

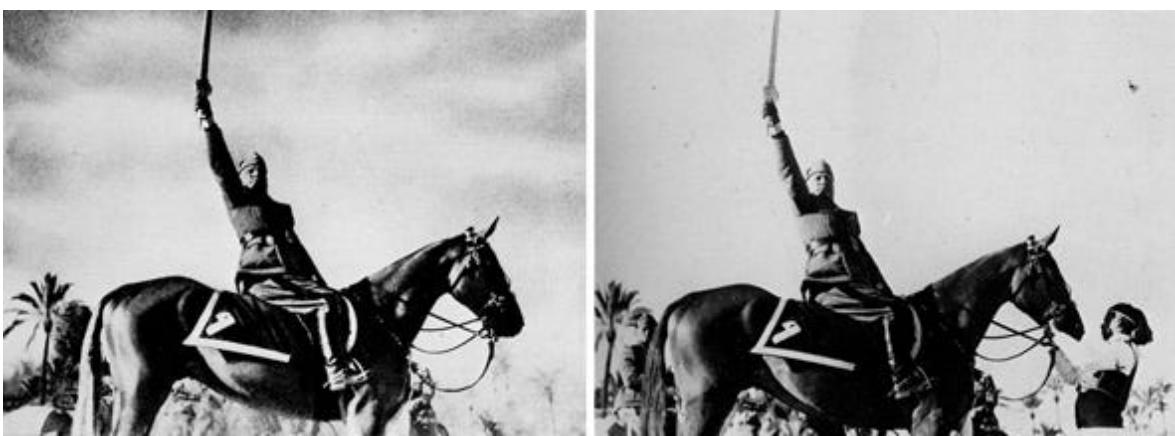


Figura 1.2: Esempio di contraffazione storica. Sulla destra l'immagine originale, sulla sinistra Mussolini senza l'aiuto dell'uomo impegnato a tenere fermo il cavallo

di valutare in che modo le performance del modello possano variare, anche sulla base della difficoltà dei campioni presenti nei due *dataset* utilizzati, applicando due tipi di ottimizzatori differenti. In un primo momento lo *Stochastic Gradient Descent* e, successivamente, *Adam*. L'ipotesi è che le prestazioni della rete in termini di precisione del riconoscimento risulteranno inferiori per il *dataset* più impegnativo. Inoltre, anche l'utilizzo di un differente ottimizzatore influenzera le performance della Rete. Quanto alle metriche utilizzate per la valutazione del modello, ci si focalizzerà sull'accuratezza del classificatore durante la fase di *testing*, analizzando anche la *confusion matrix* ottenuta, suddivisa in falsi positivi, positivi, falsi negativi e negativi. L'accuratezza è una metrica che, generalmente, descrive come il modello si comporta nelle classi tra cui apportare la classificazione. Risulta essere particolarmente utile quando le stesse hanno pari importanza. Viene calcolata come il rapporto tra le predizioni corrette e il numero totale delle predizioni da effettuare.^[3] In aggiunta, altra metrica utilizzata per effettuare un confronto, sulla base dell'utilizzo dei due ottimizzatori, è considerare il livello di generalizzazione della Rete su un set di dati completamente nuovo e sulla base dei campioni di immagine utilizzati in fase di *training*.



Figura 1.3: Esempio di contraffazione storica - Tydings con il capo del Partito Comunista Americano

1.2 Il problema delle Fake News

Il fenomeno delle *fake news* è una problematica ampiamente diffusa al giorno d'oggi. Con l'arrivo della pandemia da COVID-19 il problema è esploso, e ciò è stato favorito grazie all'utilizzo intensivo dei social che in questo caso giocano un ruolo di fondamentale importanza poiché chiunque, in qualunque momento, può accedere alle informazioni condivise in Rete. Sebbene ciò costituisca un vantaggio dal punto di vista dell'accesso all'informazione, che attraverso i social è immediata e alla portata di tutti, comporta allo stesso tempo il grande svantaggio per il quale chi, lontano dall'avere buone intenzioni, approfitta di questo potente mezzo per favorire la diffusione di bufale mediatiche, con lo scopo di creare confusione e mettendo a rischio il diritto a una corretta informazione. A questo proposito, è interessante lo studio condotto nel 2021 dall'azienda *Deloitte* [4] il quale afferma che, tra le persone partecipanti, il 23% preferisce i social come fonte primaria di notizie ma, allo stesso tempo, il 22% ha smesso di utilizzare almeno una piattaforma social. Tra le motivazioni dietro questa scelta vi è proprio la presenza eccessiva di *fake news*. Esistono degli accorgimenti da poter adottare per contrastare questo fenomeno, come ad esempio il controllare che la fonte di riferimento sia una fonte verificata e, ancora, che l'autore della notizia sia citato per poter effettuare una verifica sulla sua credibilità. Purtroppo l'adozione di queste misure non è affatto scontata dal momento che le informazioni corrono veloci e non sempre si è disposti ad impiegare il proprio tempo per mettere in atto questa serie di controlli. Inoltre, bisogna considerare la possibilità che, nonostante gli accorgimenti, ci si imbatta comunque in notizie prive di fondamento. Da ciò nasce la necessità di poter distinguere facilmente tra *fake news* e notizie aventi una base scientifica. Un ruolo fondamentale in questo campo è svolto dall'utilizzo delle immagini che

accompagnano gli articoli poiché queste, insieme ai titoli, ne rappresentano l'elemento più evidente e costituiscono quindi il primo fattore atto ad attirare l'attenzione di chi legge. La modifica delle immagini è poi resa estremamente semplice grazie alla diffusione di applicazioni disponibili, anche gratuitamente, a chiunque ne voglia usufruire, attraverso il *Web* o tramite *App Store* e simili. Tra i più famosi ricordiamo ad esempio *Adobe Photoshop*, *PicsArt Photo Studio*, etc. Sebbene la manipolazione delle immagini a scopo personale non rappresenti un problema, quando questa ha come obiettivo quello di diffondere disinformazione, rappresenta un vero e proprio pericolo. Basti pensare agli effetti che ciò comporta in ambito politico e sociale. Quanto detto finora conferma come sia cruciale avere un mezzo, laddove non sia possibile all'occhio umano distinguere tra un'immagine manipolata e una originale, che sia in grado di attuare autonomamente questa distinzione. Ciò è reso possibile grazie all'uso dell'intelligenza artificiale che, ormai da anni, mette a disposizione metodologie in grado, attraverso l'analisi delle immagini, di riconoscere le manipolazioni digitali e di conseguenza classificare, tramite l'utilizzo di appositi algoritmi, in autentici e falsi.

1.3 Campi applicativi

Per *Digital Image Forensics*(DIF) si intende un ramo di ricerca nato con lo scopo di validare l'autenticità di un'immagine recuperando informazioni basate sulla storia che si nasconde dietro essa. Questa disciplina deriva da domini di ricerca relativi alla sicurezza multimediale e sfrutta strumenti di elaborazione e analisi delle immagini per recuperare informazioni. Due sono i principali percorsi di ricerca:

- Il primo include metodi che tentano di rispondere al quesito secondo cui l'immagine catturata corrisponda effettivamente al dispositivo con cui si dichiara di averla scattata. Ciò avviene tramite una analisi balistica per identificare il dispositivo che ha catturato l'immagine o, al contrario, determinare quelli con i quali non è stata catturata procedendo dunque per esclusione.
- Il secondo gruppo di metodi invece, mira ad esporre tracce di manipolazione semantica studiando incongruenze statistiche dell'immagine. A questi metodi ci si riferisce con il nome *Image Forgery Detection*.

Altri campi nell'ambito dell'elaborazione delle immagini comprendono:

- *Image Reconstruction*: per il restauro di immagini deteriorate con lo scopo di ripristinare, anche parzialmente, il contenuto originale dell'immagine di partenza.
- *Self Embedding*: che comprendono tecniche di inserimento ed estrazione di informazioni con lo scopo di alterarne il contenuto come la rimozione di parti significative di un'immagine sostituite con altre opportunamente create.
- *Video Analysis*: analisi comportamentale, al fine di individuare comportamenti sospetti, come nel caso di furto.

- *Ricostruzione 3D*: estrazione di informazioni tridimensionali contenute in una scena al fine di ricavare misure originali.
- *Steganografia*: individuazione di informazioni nascoste all'interno di un'immagine. [5]

Sebbene questa disciplina sia abbastanza recente, è strettamente legata a diversi campi di ricerca. La DIF eredita infatti i suoi obiettivi dalla scienza forense classica e dal più recente campo dell'informatica forense. Queste, in generale, mirano ad esporre prove di reati; per farlo devono però fare i conti con le abilità di chi è intenzionato a nascondere, o falsificare, le proprie tracce rispetto al reato di contraffazione commesso. Nelle immagini digitali, sia il processo di acquisizione che le tecniche di manomissione lasciano tracce, consentendo di poter agire su queste per determinarne l'autenticità. Il compito degli esperti forensi è proprio quello di esporre queste tracce sfruttando le conoscenze esistenti sui meccanismi dietro le immagini digitali, aiutati anche dai risultati consolidati nel campo della ricerca sulla sicurezza multimediale. [6]

Capitolo 2

Stato dell'arte

2.1 Image Forgery Detection

In letteratura, l'individuazione delle alterazioni digitali subite dalle immagini è conosciuta con il nome di *Image Forgery Detection*. Quanto alle tecniche utilizzate per effettuare le modifiche, si distinguono in due categorie: attive (intrusive) e passive (non intrusive). Nel primo caso, l'immagine comprende vari tipi di pre-elaborazione come i *watermark* inclusi, come in figura 2.1 [7], o le *signature* che vengono aggiunte all'immagine di partenza. Nel caso delle

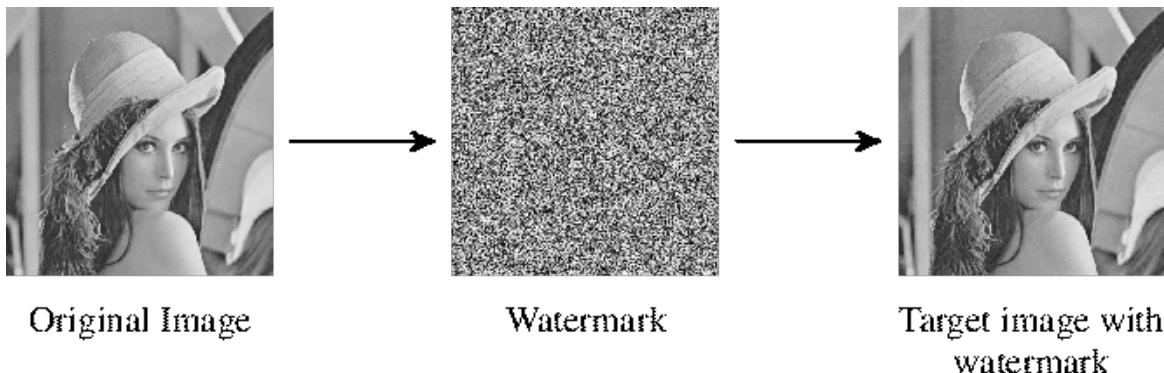


Figura 2.1: Esempio di *digital watermarking*

signature, vengono incluse tutta una serie di informazioni secondarie ottenibili dall'immagine. Tipicamente ha le seguenti proprietà:

- Solo chi invia può contrassegnare l'immagine e chi la riceve può solo effettuare la validazione
- Utenti non verificati non possono modificarne la *signature*

Se l'immagine subisce delle alterazioni, l'informazione speciale non viene estratta dall'immagine ottenuta. Quanto alle metodologie passive, queste introducono l'alterazione all'interno dell'immagine tramite una serie di trasformazioni come ad esempio le tecniche di ritocco o, ancora, *splicing*, *copy-move* e *removal*. Se il ritocco fotografico non viene considerato dannoso, in quanto l'immagine di partenza non viene modificata del tutto rispetto al suo contenuto,

queste ultime sono considerate molto più aggressive e pericolose poiché danno origine ad un'immagine completamente nuova e di conseguenza alterata rispetto a ciò che rappresentava in origine. La figura 2.2 riportata di seguito sintetizza le principali tecniche utilizzate nel campo dell'*Image Forgery*. [8]

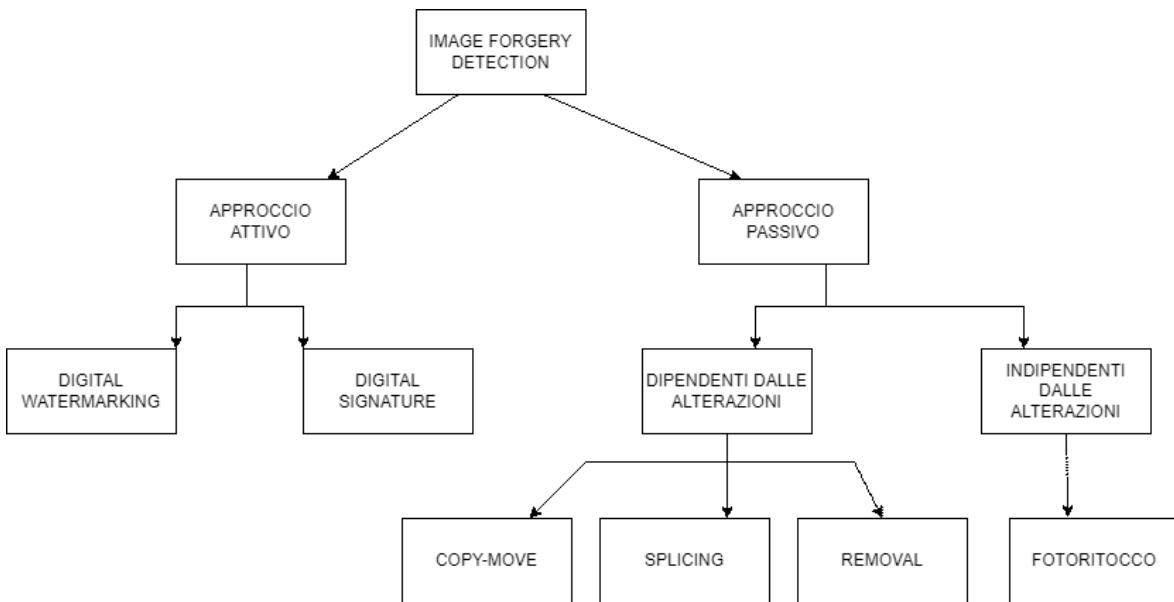
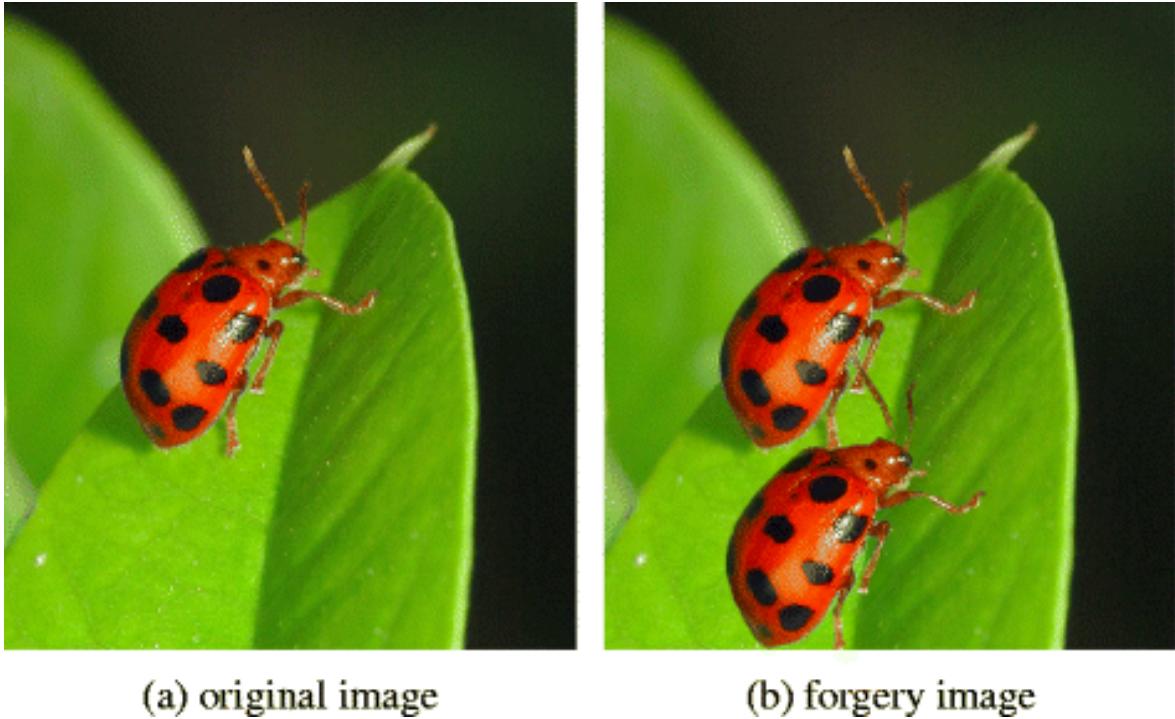


Figura 2.2: Tecniche per l'alterazione di immagini

2.1.1 Tecniche per la manipolazione delle immagini

Ai fini dell'elaborato, l'attenzione verrà posta sulle tre tecniche di manipolazione principali conosciute in letteratura: *copy-move*, *splicing* e *removal*.

- **COPY-MOVE:** ampiamente utilizzata, è una tecnica che consiste nell'estrare una specifica regione dell'immagine di partenza e copiarla all'interno della stessa, in posizione differente. Negli step successivi vengono poi applicate una serie di trasformazioni come rotazioni, compressione, sfocature etc. Lo scopo principale è quello di nascondere un dettaglio o evidenziare un particolare oggetto come mostrato in figura 2.3[9]. Le metodologie studiate per contrastare questo tipo di alterazioni sono classificabili in tre categorie: basate su *Block-generation*, basate sull'individuazione di *Keypoint* e Varie. Nel primo caso si divide l'immagine in blocchi di pixel sovrapposti o meno e, successivamente, questi blocchi vengono processati tramite SVD, DFT, etc, usati per generare il vettore delle *feature* da dare in input all'algoritmo SVM che ne effettua la classificazione. Nel caso dell'individuazione di *Keypoint*, si utilizzano algoritmi quali SIFT o SURF. I *Keypoint* sono punti speciali all'interno dell'immagine che non cambiano anche subendo trasformazioni quali traslazione e ridimensionamento. Ci sono poi varie tecniche ibride basate sul *Deep Learning* che si sono dimostrate efficaci sia dal punto di vista delle performance che sulla ridotta complessità computazionale. [10]

Figura 2.3: Esempio applicazione tecnica *copy-move*

- **SPLICING:** questo tipo di tecnica, a differenza della precedente, implica l'aggiunta di una particolare regione di immagine su una differente. Un esempio di applicazione di questa tecnica è mostrato in figura 2.4 [11]. Questa particolare metodologia, a differenza del metodo *copy-move*, richiede l'utilizzo di più approcci per la risoluzione del problema ed è per questo più complessa nella gestione. Infatti, una singola immagine può risultare dalla combinazione di due o più immagini, con conseguente creazione di molteplici inconsistenze, visibili o meno.

Figura 2.4: Esempio applicazione tecnica *splicing*

- **REMOVAL:** in questo caso, una porzione di immagine viene rimossa e, successivamente, la parte rimossa viene riempita in modo tale da rendere impercettibile la modifica effettuata, come mostrato in figura 2.5. [12]

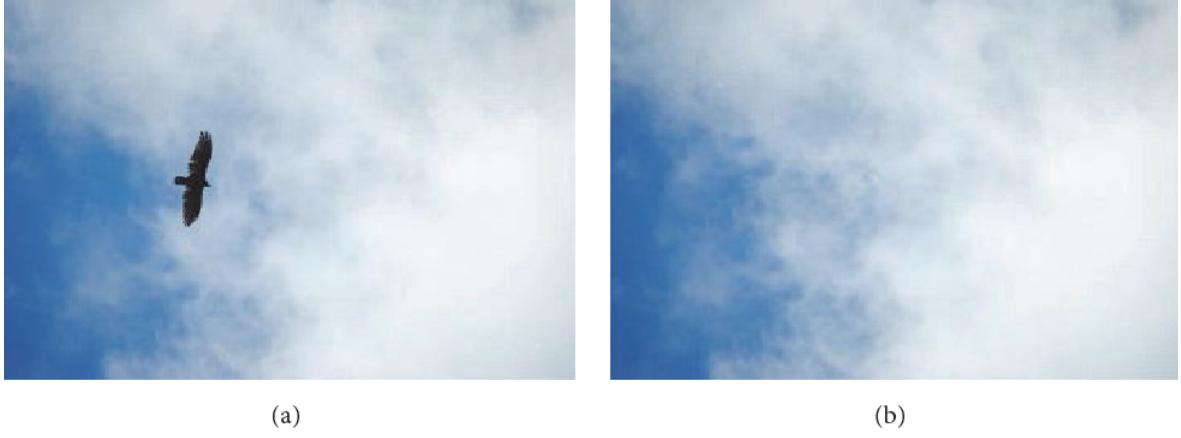


FIGURE 1: (a) Original image. (b) Forged image.

Figura 2.5: Esempio applicazione tecnica *removal*

2.1.2 Reti Neurali Convoluzionali e Transfer Learning

Tra le tecniche ampiamente utilizzate in letteratura nel campo dell' *Image Forgery Detection* vi sono l'utilizzo di tecniche di *Deep Learning*, con cui si fa riferimento alle Reti Neurali a più livelli, capaci di gestire un'enorme quantità di dati e il *Transfer Learning*. Il loro utilizzo è estremamente popolare nella risoluzione di problemi di *Machine Learning*, specialmente nei campi del riconoscimento delle immagini, *Computer Vision*, etc. [13] Il *Transfer Learning*, in particolare, è una tecnica molto utilizzata nel campo della classificazione delle immagini. Si basa sulla possibilità di riutilizzare un modello precedentemente allenato su una determinata problematica, per risolverne un'altra (ma simile) sulla base della conoscenza ottenuta. Ciò significa che l'allenamento della Rete non dovrà essere ripetuto da zero per ogni nuovo problema considerato, ma ci si potrà concentrare sull'addestramento degli ultimi livelli dedicati alla classificazione, basandosi sulle *feature* ottenute dagli strati precedenti. Questo rappresenta un grande vantaggio che consente di risparmiare risorse poiché l'elaborazione verrà limitata ad un numero notevolmente minore di parametri, e, di conseguenza, tempo. Infatti, il *Transfer Learning* è spesso utilizzato quando la fase di *training* per un determinato *task* richiede un eccessivo utilizzo di risorse. Tra gli altri vantaggi, vi è la possibilità di utilizzare un set di dati ridotto riuscendo comunque ad ottenere ottimi risultati e ciò è estremamente vantaggioso, soprattutto se consideriamo che i set di dati sono costosi da ottenere. Sappiamo infatti che, normalmente, per ottenere buoni risultati è necessario allenare una Rete con un notevole numero di dati e non è detto che questi siano sempre disponibili in grandi quantità. Punto chiave di questa tecnica è la generalizzazione. Ciò significa che solo la conoscenza che può essere utilizzata da un modello su scenari e condizioni diverse viene "trasferita". Questo consente di avere un'elevata flessibilità in quanto i modelli sviluppati con questa metodologia possono essere utilizzati a prescindere dalle condizioni e su *dataset* diversi. [14] Sono due le metodologie per riutilizzare un modello:

- Estrazione delle *feature*: dove non sarà necessario allenare da zero l'intero modello in quanto gli strati di convoluzione avranno già estratto tutte le *feature* necessarie per la

classificazione. Occorrerà lavorare solo sul classificatore.

- *Fine Tuning*: limita la computazione ad un numero inferiore di parametri. Ciò avviene "sbloccando" solo alcuni dei livelli superiori della Rete pre-addestrata così da allenare in maniera più specifica il modello. Pertanto, si perfeziona l'estrazione delle *feature* presenti nel modello di base in modo tale da renderle più adatte al nuovo problema. Ciò non solo consente di velocizzare i tempi di addestramento, ma anche di aumentare l'accuratezza della Rete. [15]

L'architettura di una Rete Neurale Convoluzionale (CNN) è simile a quella del modello di connettività dei neuroni nel cervello umano, a cui si ispira. Tramite una CNN si è in grado di classificare cosa un'immagine rappresenta e di identificarne il contenuto. Quanto all'architettura di una CNN, tipicamente consta di diversi livelli che partono dal livello di Input costituito dall'immagine da analizzare, segue il livello di Convoluzione il cui obiettivo è quello di individuare particolari *feature* all'interno dell'immagine come ad esempio curve, angoli, etc. Con l'aumentare di questi livelli si riesce ad individuare forme sempre più complesse. Ogni livello di Convoluzione è solitamente seguito da un livello ReLU (Rectified Linear Unit), il cui scopo è quello di annullare tutti i valori minori di zero derivati dal livello precedente. Segue il livello di Pooling, che permette di ridurre ulteriormente la dimensione dell'immagine in modo tale che sia più semplice da processare senza però incorrere nella perdita di *feature*, fondamentali per una corretta predizione. Infine vi è il livello Fully Connected il quale genera un vettore N-dimensionale, dove N sta a rappresentare il numero di classi tra cui bisognerà effettuare la classificazione. Questo livello comprende un numero elevato di parametri che richiedono una complessa computazione e molto tempo in fase di *training*. [13] Un tipico esempio di architettura di una Rete Neurale Convoluzionale è mostrato in figura 2.6.

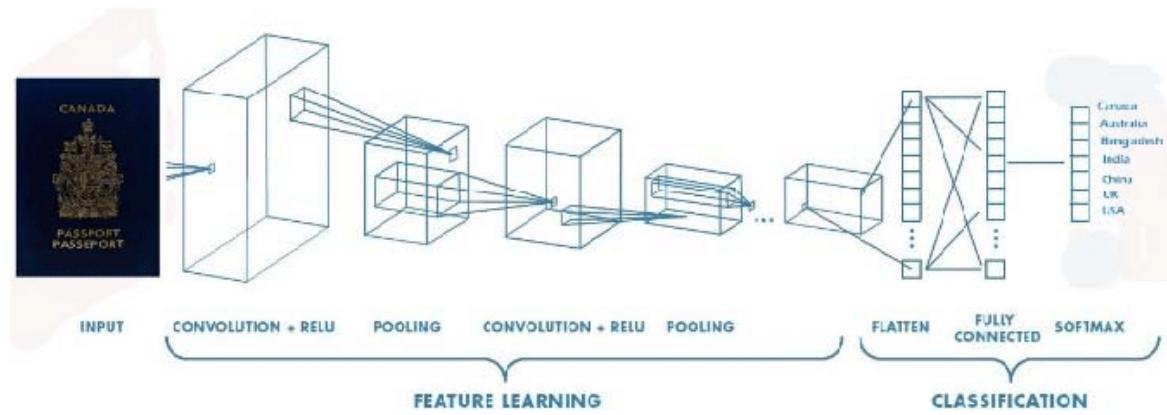


Figura 2.6: Architettura di una CNN

2.1.3 Risultati ottenuti

I risultati ottenuti in letteratura, attraverso le metodologie precedentemente citate, vengono riportati di seguito. In particolare, quanto all'utilizzo delle Reti Neurali Convoluzionali, viene proposto un modello che combina l'utilizzo delle CNN, l'*Error Level Analysis* (ELA) e dello *Sharpening Filter*.

- **ELA:** è una tecnica forense per la rilevazione di immagini falsificate. Si basa sulla compressione dei dati con perdita. Prevede di comprimere in maniera irreversibile un'immagine così da ridurne la dimensione con conseguente perdita di parte dell'informazione originale. Quando un'immagine subisce, ad esempio, la compressione JPEG, il livello di compressione di ogni pixel è lo stesso. L'algoritmo lavora su immagini viste come griglie e suddivise in blocchi di 8x8 pixel. Per immagini che non abbiano subito un qualche tipo di alterazione digitale, il livello di compressione delle griglie dovrebbe essere lo stesso. Al contrario, per quelle modificate, il livello di compressione risulterà essere differente. Ci sarà quindi una certa incoerenza tra le griglie. L>Error Level Analysis quindi, salva l'immagine con un livello di compressione pari al 95%. Successivamente, valuta la differenza tra l'immagine di partenza e quella appena salvata. Se l'immagine non ha subito alterazioni, la differenza sarà la stessa per tutte le griglie di pixel. Tuttavia, se l'immagine risulta avere delle incoerenze, la differenza non sarà la stessa. Questa contraddizione nel livello di compressione mostrerà le aree sospette nell'aver subito delle modifiche come mostrato in figura 2.7. La figura mostra il risultato dell'applicazione dell'ELA su un'immagine modificata. I bordi bluastri rivelano la presenza di una differenza nel livello di compressione nell'immagine. È possibile vedere come la forma dell'animale, delimitata dai bordi blu, sia in realtà il risultato di un'alterazione e quindi, appartenente ad un'altra immagine. [16] Occorre sottolineare che questo metodo funziona bene solo con immagini che abbiano subito un qualche tipo di compressione.



Figura 2.7: Error Level Analysis

- **Sharpen Filter:** allo scopo di incrementare ulteriormente l'accuratezza del modello per rilevare eventuali alterazioni, si è deciso di combinare l'ELA con lo Sharpen Filter. Per *Sharpening* si intende il processo con cui si aumenta il contrasto tra le regioni chiare e scure di un'immagine così da esaltare le *feature* della stessa. L'utilizzo di questo filtro sull'immagine modificata presenterà un effetto contrasto non uniforme in quanto bordi e linee dell'immagine risulteranno essere sfocati o distorti. [17]

Quanto alla seconda metodologia considerata, il *Transfer Learning*, i modelli considerati sono il VGG16 e Resnet50 con alcune loro variazioni.

- **VGG16:** abbreviazione che sta per *Visual Geometry Group*, ossia il gruppo di ricercatori che l'ha sviluppata, mentre il 16 implica che questo tipo di architettura consta di 16 strati, 13 di Convoluzione e 3 *Fully Connected*, come mostrato in figura 2.8. I 16 strati sono meglio descritti di seguito:

1. Convoluzione con l'utilizzo di 64 filtri
2. Convoluzione con l'utilizzo di 64 filtri e Max-pooling
3. Convoluzione con l'utilizzo di 128 filtri
4. Convoluzione con l'utilizzo di 128 filtri e Max-pooling
5. Convoluzione con l'utilizzo di 256 filtri
6. Convoluzione con l'utilizzo di 256 filtri
7. Convoluzione con l'utilizzo di 256 filtri e Max-pooling
8. Convoluzione con l'utilizzo di 512 filtri
9. Convoluzione con l'utilizzo di 512 filtri
10. Convoluzione con l'utilizzo di 512 filtri e Max-pooling
11. Convoluzione con l'utilizzo di 512 filtri
12. Convoluzione con l'utilizzo di 512 filtri
13. Convoluzione con l'utilizzo di 512 filtri e Max-pooling
14. Fully Connected con 4096 nodi
15. Fully Connected con 4096 nodi
16. Strato di Output con funzione di attivazione Softmax e 1000 nodi

Gli svantaggi legati a questo modello sono principalmente due:

- Richiede tempi di allenamento molto lunghi.
- I pesi della Rete sono molto grandi.

D'altra parte i vantaggi:

- Facile da implementare e costituisce un ottimo punto di partenza.

- L'utilizzo di un *dataset* ridotto è possibile, applicando tecniche di *data augmentation* per contrastare il fenomeno dell'*overfitting*, riuscendo ad ottenere buoni risultati.
- Si può applicare il Fine Tuning così da incrementarne le performance. [18]

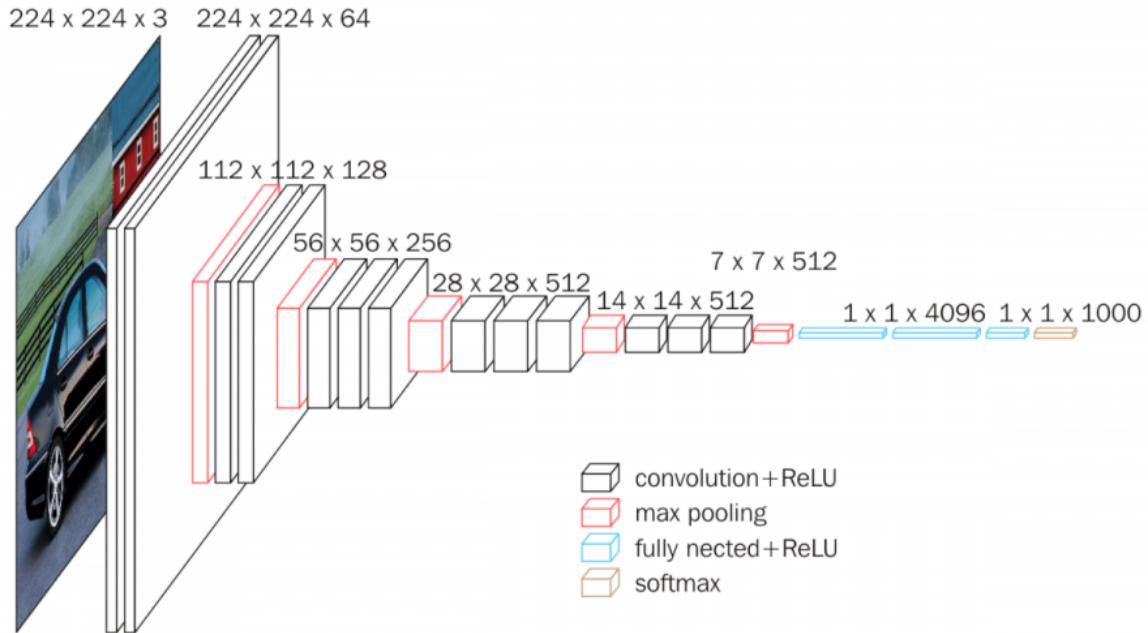


Figura 2.8: Architettura VGG16

- **ResNet50:** sta per *Residual Network* ed è un modello che, come suggerito dal nome, consta di 50 strati. Mentre il numero di strati sovrapposti può arricchire le *feature* del modello, una Rete più profonda può presentare il problema per cui l'errore, in fase di addestramento (con conseguente perdita di precisione), aumenti invece che diminuire. Si tratta dunque di un problema di ottimizzazione e degradazione. I modelli più profondi risultano essere difficili da ottimizzare, la precisione si satura e da ciò si ha l'aumento dell'errore. La ResNet nasce con lo scopo di risolvere questo problema e per farlo utilizza i blocchi residui. Questi consentono, nel peggio dei casi, ai livelli più profondi della Rete di non apprendere nulla senza però pesare sulle prestazioni. Nella maggior parte dei casi invece, questi livelli riusciranno ad apprendere qualcosa che aiuterà a migliorarne le prestazioni. La ResNet 50 utilizza una struttura a collo di bottiglia, per evitare tempi di allenamento eccessivi, utilizzando prima una convoluzione 1x1 per ridurre le dimensioni delle *feature map*, quindi una convoluzione 3x3 e infine 1x1 per ripristinare la dimensione originale. Ciò equivale a ridurre il numero di parametri per lo stesso numero di livelli. Così facendo si è riusciti a ridurre la complessità di calcolo, limitando il tasso di errore e migliorando le prestazioni. [19]

Nella tabella riportata di seguito sono mostrati i risultati ottenuti dai modelli precedentemente descritti e presenti in letteratura, espressi in termini di accuratezza in fase di *testing*.

Tabella 2.1: Risultati ottenuti in letteratura sul *dataset* CASIA V2.0 per il problema dell'*Image Forgery*

MODELLO	TRAINING ACCURACY (%)	VALIDATION ACCURACY (%)	TRAINING LOSS (%)	VALIDATION LOSS (%)
CNN	77.13	75.68	0.46	0.51
CNN-ELA	94.00	90.02	0.21	0.30
CNN-Sharpen-ELA	97.00	94.52	0.10	0.29
VGG16	80.25	81.92	0.51	0.49
VGG16-Bottleneck	83.18	84.26	0.32	0.38
VGG16-FineTuned	99.16	94.77	0.01	0.30
ResNet50-FineTuned	98.65	95.72	0.04	0.18

Capitolo 3

Approccio tecnico

3.1 Architettura della CNN proposta

Il modello proposto è basato sull’architettura presentata da Y.Rao et al [20] ed implementata da Kyriakos Psarakis et al.[21]. L’architettura della Rete proposta consta di nove strati di Convoluzione e due di Max-Pooling come mostrato in figura 3.1. Come input alla Rete viene passata una *patch* avente dimensione 128x128x3, dove 3 rappresenta i canali RGB. I primi due strati di Convoluzione hanno 30 *kernel* di dimensione 5x5 mentre gli altri, 16 *kernel* di 3x3 con lo scopo di estrarre le *feature* necessarie. Seguono operazioni di Pooling. Ogni strato di Convoluzione utilizza la ReLU come funzione di attivazione. Bisogna inoltre aggiungere che per migliorare la generalizzazione, viene applicata la *Local Response Normalization* (LRN). Questa consiste in un livello che implementa il concetto conosciuto in neurobiologia con il nome di inibizione laterale secondo cui un neurone ha la capacità di ridurre l’attività dei suoi vicini. Questo strato è utile quando si ha a che fare con i neuroni derivati dalla ReLU che hanno attivazioni illimitate, il compito della *Local Response Normalization* è quello di normalizzare questo fenomeno. L’obiettivo è quello di migliorare il contrasto tra i pixel, attraverso un valore di massimo locale, in modo tale da eccitare i neuroni successivi. [22] Ogni strato di Convoluzione, a parte il secondo, è poi inizializzato utilizzando l’inizializzazione di *Xavier*. La ragione principale è che questo evita di avere valori troppo alti o prossimi allo zero. Ciò è consentito mantenendo lo stesso valore della varianza per tutti gli strati. Quanto ai *kernel* del secondo livello di Convoluzione, vengono inizializzati tramite l’uso di 30 filtri *SRM high-pass*. I filtri *SRM* permettono di individuare cambiamenti sui margini della regione di immagine. Queste variazioni potrebbero infatti significare che l’immagine sia contraffatta, in quanto i bordi mostreranno inconsistenze che non si integrano bene col resto dell’immagine. Come risultato, il cambiamento tra un pixel e un altro risulterà drastico e, quindi, il modello potrà meglio interpretarlo come dimostrazione di appartenenza ad un’immagine alterata.

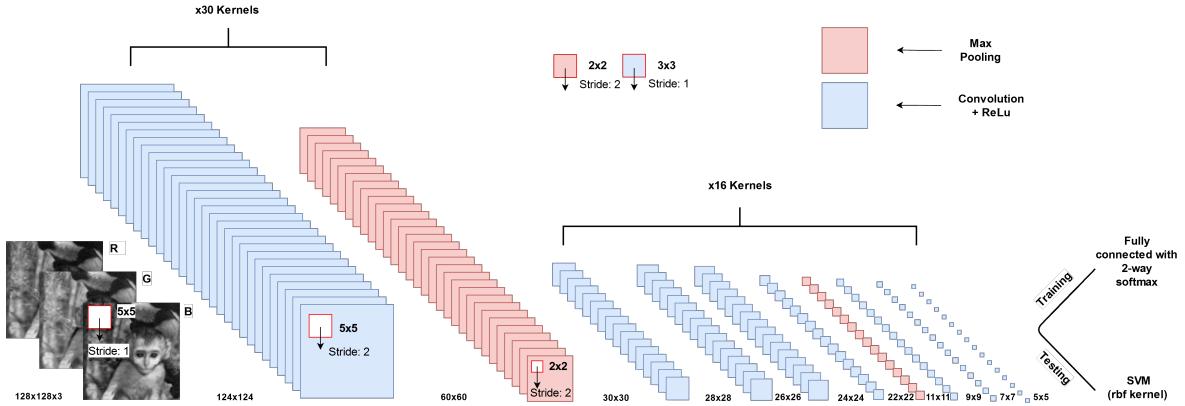


Figura 3.1: Architettura della CNN utilizzata

3.1.1 Il problema dell'overfitting

Tra le problematiche che è possibile incontrare durante la fase di allenamento del modello vi è l'*overfitting*. Questo problema si verifica quando un modello si adatta esattamente ai suoi dati di *training* e se ciò accade, sfortunatamente, l'algoritmo non riesce a funzionare in modo accurato su dati che non conosce, vanificando il suo scopo. Quando il modello viene allenato troppo a lungo su determinati campioni, o è troppo complesso, può cominciare ad apprendere informazioni non rilevanti all'interno del *dataset*. Se l'eccessivo allenamento o la complessità del modello si traducono in *overfitting*, una risposta logica potrebbe essere quella di sospendere anticipatamente il processo di addestramento, noto come *early stopping*, o di ridurne la complessità. Tuttavia, se ci si ferma troppo presto o si escludono troppe funzionalità, ci si potrebbe imbattere nel problema opposto, conosciuto con il nome di *underfitting*. Si verifica quando il modello non viene addestrato per un tempo sufficiente e le variabili di input non sono sufficientemente significative per determinare una relazione tra le variabili di input e quelle di output. In entrambi i casi, il modello non sarà in grado di generalizzare, come mostrato in figura 3.2 [23]. In letteratura esistono diverse metodologie per evitare il problema dell' *overfitting*:

- *Early stopping*: come menzionato in precedenza, questo metodo ha lo scopo di fermare l'allenamento prematuramente. Il rischio è che ciò avvenga troppo presto con conseguente ricaduta nel problema dell'*underfitting*.
- Allenare il modello con un numero maggiore di campioni: aumentare il set di dati per l'allenamento potrebbe incrementarne l'accuratezza.
- *Data augmentation*: è un modo per avere a disposizione un numero di campioni maggiore rispetto al dataset di partenza, senza effettivamente raccogliere nuovi elementi. Per fare ciò è possibile introdurre rotazioni alle immagini.
- Selezione delle *feature*: è il processo di identificazione delle *feature* più pertinenti all'interno dei dati per il *training*, eliminandone quelle ridondanti.

- Regolarizzazione: se si verifica il problema dell'*overfitting* quando il modello è troppo complesso, ha senso ridurre il numero di *feature*. Questo è possibile se si conoscono in anticipo gli input da eliminare. Quando ciò non è possibile, interviene il metodo di regolarizzazione. Questo metodo applica una penalità ai parametri di input che hanno coefficienti molto grandi. [24]

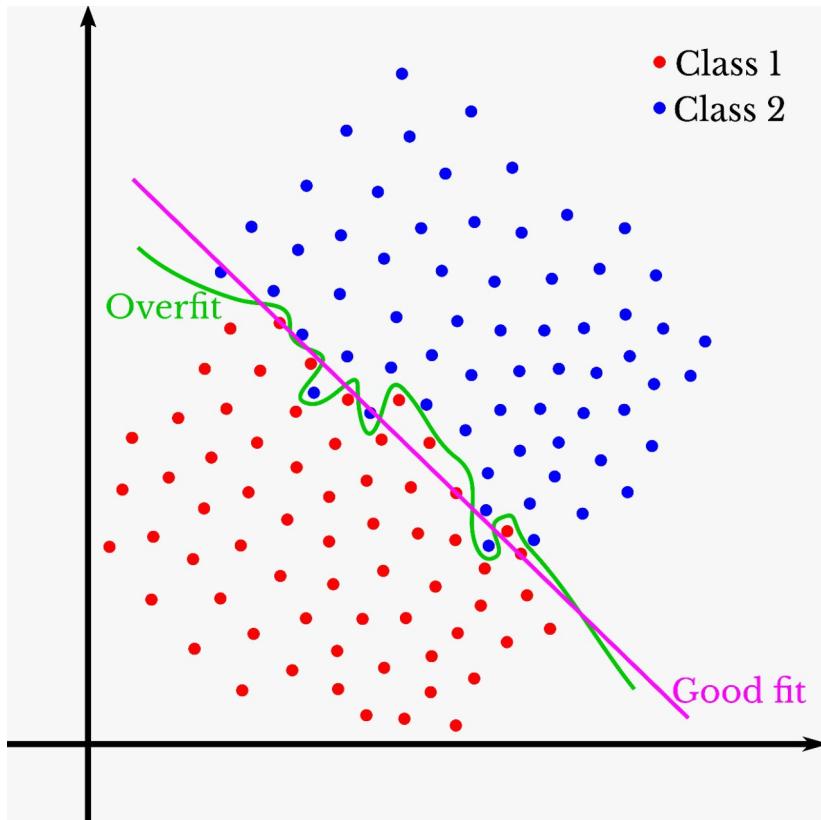


Figura 3.2: Esempio grafico del fenomeno dell'*overfitting*

3.1.2 Training

I principali step per l’allenamento della Rete sono due:

- CNN *training*
- SVM *training*

Nel primo passaggio, per poter allenare l’architettura della CNN descritta precedentemente, in modo tale che possa concentrarsi sul riconoscimento degli artefatti, bisogna effettuare un processo di estrazione delle *patch* dalle immagini presenti nei *dataset*. La dimensione delle *patch* estratte è 128x128x3, ciò sta a significare che vi sarà una *patch* per ogni canale di colore. L’estrazione viene effettuata applicando una finestra scorrevole della dimensione della *patch* con *stride* pari ad otto per l’intera immagine. Successivamente, le *patch* appartenenti alle immagini alterate vengono distinte rispetto a quelle non alterate:

- Per quanto riguarda le *patch* manomesse, ognuna di queste viene comparata con la *patch* equivalente della maschera (appartenente alla stessa regione di immagine che si

sta considerando). Di queste, si conservano solo quelle aventi regioni sospette dell'aver subito alterazioni, come mostrato in figura 3.3. [21] Tuttavia, quanto al numero di *patch* estratte da mantenere, se ne conservano due per ogni immagine. La motivazione dietro questa scelta è da ricercarsi nel fatto che allenare la Rete con un numero eccessivo di campioni sarebbe computazionalmente oneroso.

- Quanto alle *patch* appartenenti ad immagini autentiche, viene applicata la stessa tecnica descritta in precedenza. L'unica differenza sta nel fatto che le *patch* da tenere vengono selezionate in maniera randomica.

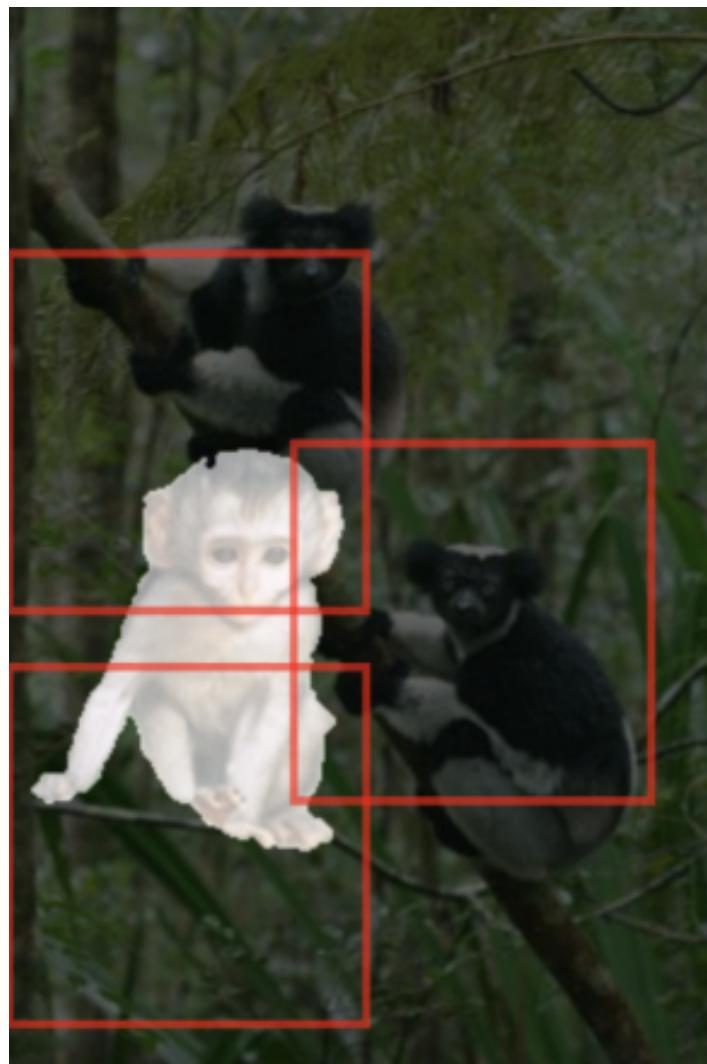


Figura 3.3: Estrazione delle *patch*

Infine, per migliorare l'abilità di generalizzare della CNN, ed evitare il problema dell'*overfitting*, viene adottata la tecnica conosciuta con il nome di *data augmentation*, effettuando una rotazione di 90 gradi per ogni *patch* estratta. Successivamente, le *patch* vengono date in pasto alla CNN che ne estrae un vettore 400-D per la rappresentazione delle *feature*. Le *feature* così estratte, vengono processate dal livello *Fully-Connected* attraverso l'utilizzo della funzione *softmax*. Questa funzione trasforma un vettore di K valori reali in un vettore di K valori

reali la cui somma è 1. I valori di input possono essere valori positivi, negativi, il valore zero o possono essere maggiori di uno. La *softmax* li trasforma in valori compresi tra 0 e 1, in modo che possano essere interpretati come probabilità. Se uno degli input è molto piccolo o negativo, allora ne conseguirà una probabilità piccola, al contrario, se l'input è grande, si tradurrà in una probabilità maggiore ma sempre compresa tra 0 e 1. [25] Altro metodo mirato a ridurre il problema dell'*overfitting*, consiste nell'adoperare la tecnica del *dropout*. Molto spesso risulta necessaria la sua applicazione in quanto, nel momento in cui un livello *Fully-Connected* presenta un numero elevato di neuroni, è molto probabile che si verifichi un coadattamento. Per coadattamento si intende la possibilità di estrazione delle stesse *feature* dai dati in input, o comunque *feature* molto simili tra loro, da parte di più neuroni della Rete. Questo può verificarsi quando i pesi di due neuroni connessi sono quasi identici. Il verificarsi di questa possibilità comporta due tipi di problematiche:

- Spreco di risorse dovuto alla computazione di uno stesso output.
- Se più neuroni estraggono le medesime *feature*, la rete incorrerà nell'*overfitting*.

Il *dropout* aiuta a limitare questo fenomeno. Il modo con cui questo avviene consiste nel deattivare (porre a zero) una parte di neuroni appartenenti ad un determinato livello (in questo caso quello *Fully-Connected*), ad ogni step di allenamento della Rete [26], con una probabilità del 50%. Nel secondo passaggio, ossia l'*SVM training*, l'obiettivo è quello di allenare il classificatore tramite il vettore delle *feature* ottenuto dallo step precedente. Per SVM (*Support Vector Machine*), si intende un insieme di metodi di apprendimento supervisionato utilizzati per la classificazione. L'obiettivo dell'algoritmo è quello di individuare un iperpiano capace di suddividere i set di dati in due classi, come mostrato in figura 3.4 [27]. I punti più

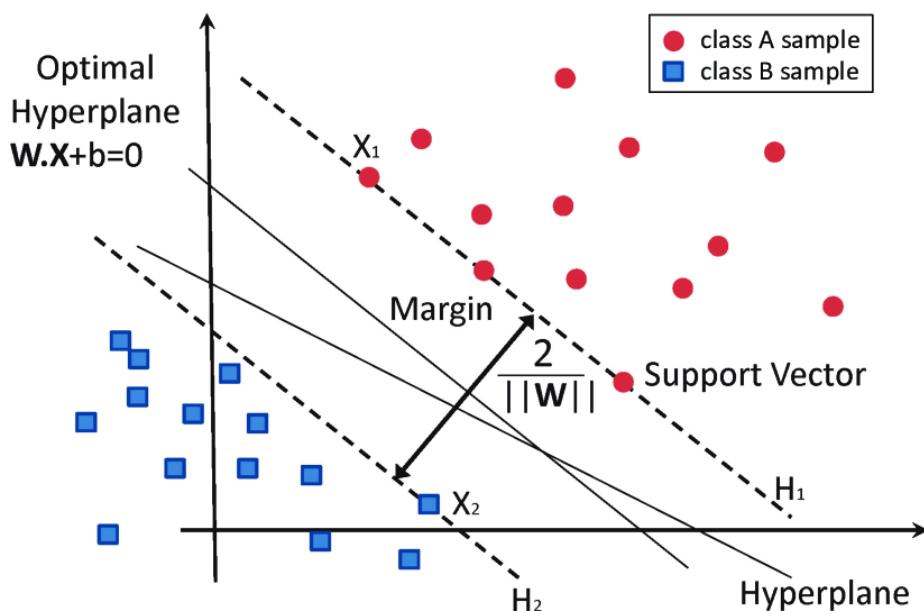


Figura 3.4: Algoritmo SVM

vicini all'iperpiano fanno sì che sia più probabile che una minuscola variazione dell'iperpiano possa modificarne la classificazione. Questi punti così descritti vengono chiamati *support*

vector. Quanto ai punti più distanti dall'iperpiano, avranno una probabilità più alta di essere classificati in maniera corretta dall'algoritmo. La distanza tra l'iperpiano e i *support vector* individuati è detta margine. Maggiore è questa distanza, migliore sarà la capacità dell'algoritmo di generalizzare e classificare correttamente i nuovi dati. [28] Infine, per effettuare una valutazione del modello citato su un numero limitato di campioni, viene utilizzata la metodologia denominata *k-Fold Cross-Validation*. Questa procedura ha un parametro chiamato *k* che si riferisce al numero di gruppi in cui deve essere suddiviso un dato campione di dati. Viene utilizzata nell'apprendimento automatico per stimare l'abilità del modello su dati mai visti prima. Vale a dire utilizzare un campione limitato per stimare le prestazioni attese del modello quando utilizzato per fare previsioni su dati non sfruttati durante la fase di addestramento. Consente di produrre una stima meno ottimistica dell'abilità del modello rispetto ad altri metodi, come una semplice suddivisione in *train set* e *test set*. [29]

3.1.3 Impatto sulle prestazioni: SGD vs Adam Optimizer

È noto come l'utilizzo di un ottimizzatore rispetto ad un altro possa significativamente influenzare le prestazioni del modello. Dunque, una volta stabilita la struttura generale della Rete, se si desidera ottenere prestazioni migliori, può essere opportuno testare un altro tipo di ottimizzatore. Ciò rappresenta una classica messa a punto degli iperparametri per provare a vedere cosa funziona meglio e meglio si adatta alla problematica trattata. Fare questo, a volte, può richiedere molte regolazioni degli iperparametri come ad esempio il *learning rate*. Un ottimizzatore è una funzione matematica utilizzata per modificare i pesi della rete sulla base dei gradienti e altre informazioni aggiuntive, a seconda della formulazione. I migliori si concentrano sull'essere più veloci ed efficienti, ma sono anche noti per la capacità di generalizzare meglio rispetto ad altri. Oggetto della fase centrale dello studio sono stati:

- Stochastic Gradient Descent (SGD)
- Adam

Il primo è un algoritmo molto popolare tra gli algoritmi di *Machine Learning*. Viene computato tenendo conto di tutti i campioni selezionati per l'addestramento, con un gradiente che viene calcolato prendendo, in maniera iterativa, un singolo campione alla volta fino a quando non vengono esaminati tutti. I parametri su cui è possibile agire per migliorare le prestazioni sono due:

- *Learning rate*: controlla la taglia dell'aggiornamento lungo il gradiente. In figura 3.5, questo sta a significare che si desidera calcolare dove debba trovarsi il punto negli step successivi (non continui). La durata di questi step rappresenta proprio il *learning rate*. La scelta di questo parametro è fondamentale quando si allena una Rete Neurale. Se il valore è piccolo, possiamo aspettarci di fare progressi costanti ma molto piccoli. Il rischio, in questo caso, è che ci si blocchi ad un minimo locale e non globale. Un passo maggiore significherebbe modificare maggiormente i pesi ad ogni iterazione, in modo che possano raggiungere il loro valore ottimale più velocemente ma, contemporaneamente,

potrebbero mancare il valore ottimo esatto. Al contrario, passi più piccoli indicano che i pesi vengono modificati di meno ad ogni iterazione e che quindi potrebbero essere necessarie più epoche per raggiungere il loro valore ottimale, ma è meno probabile che manchino l'ottimo della *loss function*.

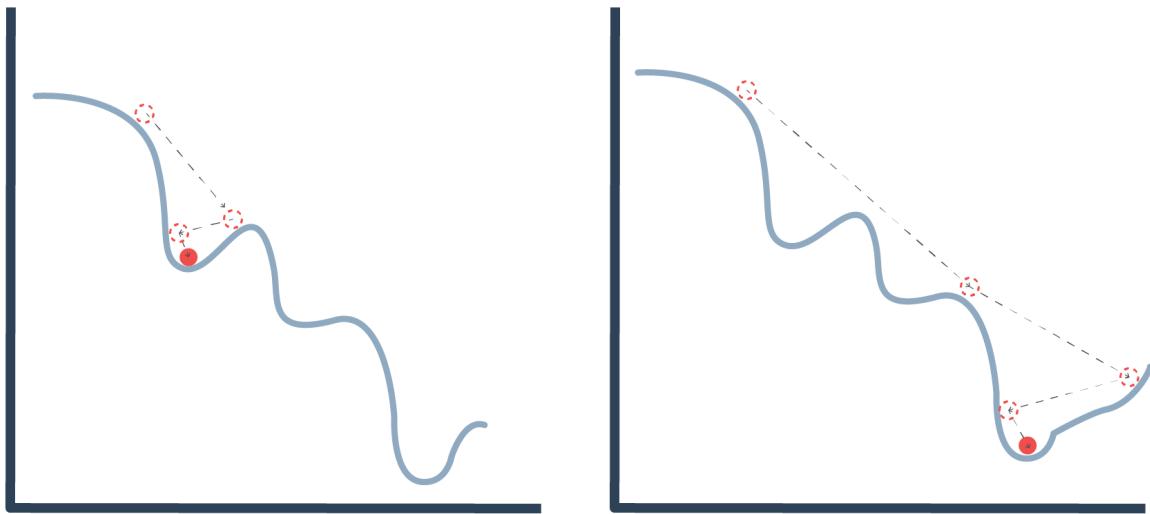


Figura 3.5: SGD: *learning rate*

- Momentum: consente l'accelerazione dell'algoritmo e ne smorza le oscillazioni. Ne risulta una convergenza più veloce e un'oscillazione ridotta. Il punto individuato dal cerchio tratteggiato in rosso in figura 3.6 accumula *momentum* mentre si muove verso il basso, diventando sempre più veloce lungo il percorso. Se non utilizzassimo il *momentum*, non si avrebbe nessuna informazione su dove il cerchio fosse prima di ogni step precedente. Dunque, ogni nuovo calcolo verrebbe basato solo sul gradiente, senza considerarne la cronologia degli step effettuati prima. Il *momentum* aiuta l'ottimizzatore a non rimanere bloccato in un punto di minimo locale. [30]

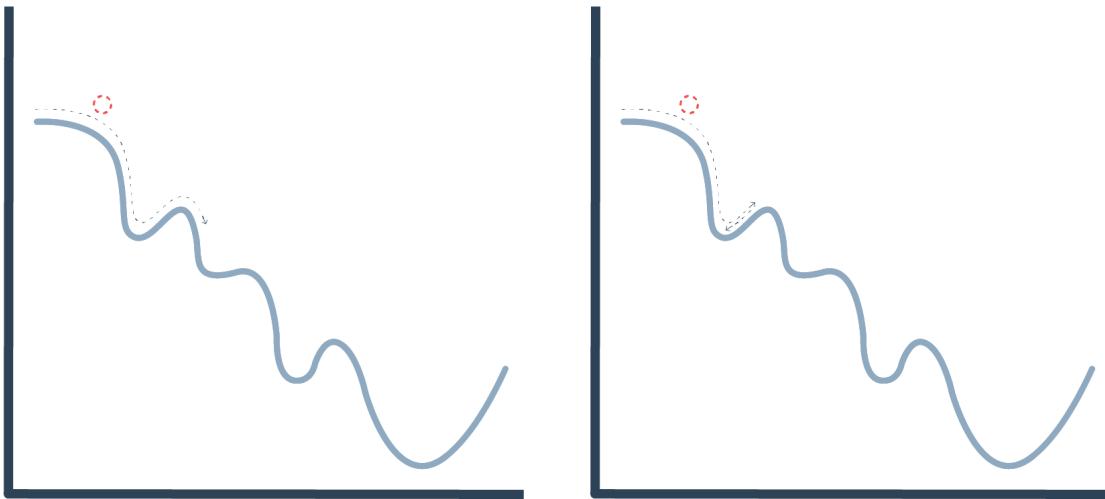
I vantaggi nell'uso di questa tecnica di ottimizzazione sono:

- Efficienza.
- Facilità di implementazione.

Tra gli svantaggi troviamo invece:

- Viene richiesto l'utilizzo di iperparametri quali numero di iterazioni e un parametro per la regolarizzazione.
- È sensibile al ridimensionamento delle *feature*. [31]

Quanto al secondo, è un algoritmo di ottimizzazione che utilizza un *learning rate* flessibile dove per flessibile si intende che calcola individualmente il *learning rate* per diversi parametri.

Figura 3.6: SGD: *momentum*

Fece la sua prima comparsa nel 2014 e può essere visto come la combinazione dell’RMSprop e dell’SGD con il *momentum*. [32] Questo metodo è molto efficiente quando si lavora con un problema che include un grande numero di dati e parametri, richiedendo meno memoria. Per fare ciò viene utilizzata la combinazione di due metodologie di discesa del gradiente:

- Momentum, come illustrato precedentemente.
- RMSprop: utilizza i gradienti quadrati per scalare il *learning rate*.

Tra le sue proprietà possiamo considerare il fatto che la regola dell’aggiornamento degli step è indipendente dall’entità del gradiente, il che aiuta molto quando si attraversano aree con gradienti molto piccoli. In questo caso, ad esempio, l’SGD fatica a navigare velocemente attraverso queste aree. Tra i vantaggi dell’utilizzo di questo ottimizzatore abbiamo:

- Efficiente in termini di computazione.
- L’utilizzo di memoria è ridotto. [33]

Appena introdotto, i risultati mostrarono come avesse prestazioni notevoli in termini di velocità di allenamento. Ciò nonostante, in letteratura sono presenti opinioni contrastanti in merito. In particolare, si credeva che lo *Stochastic Gradient Descent* fosse in grado di generalizzare meglio rispetto ad Adam, mentre quest’ultimo fosse capace di convergere più velocemente. In realtà, studi recenti hanno dimostrato come la scelta degli iperparametri possa essere il motivo per cui l’algoritmo non riesca a generalizzare bene e come, con una corretta messa a punto di questi, sia più veloce dell’SGD e non inferiore a quest’ultimo in termini di generalizzazione. [34] In figura 3.7 è mostrato un grafico rappresentante le prestazioni, in termini di *testing accuracy*, ottenute dai vari tipi di ottimizzatori, tra cui l’SGD e Adam. [35].

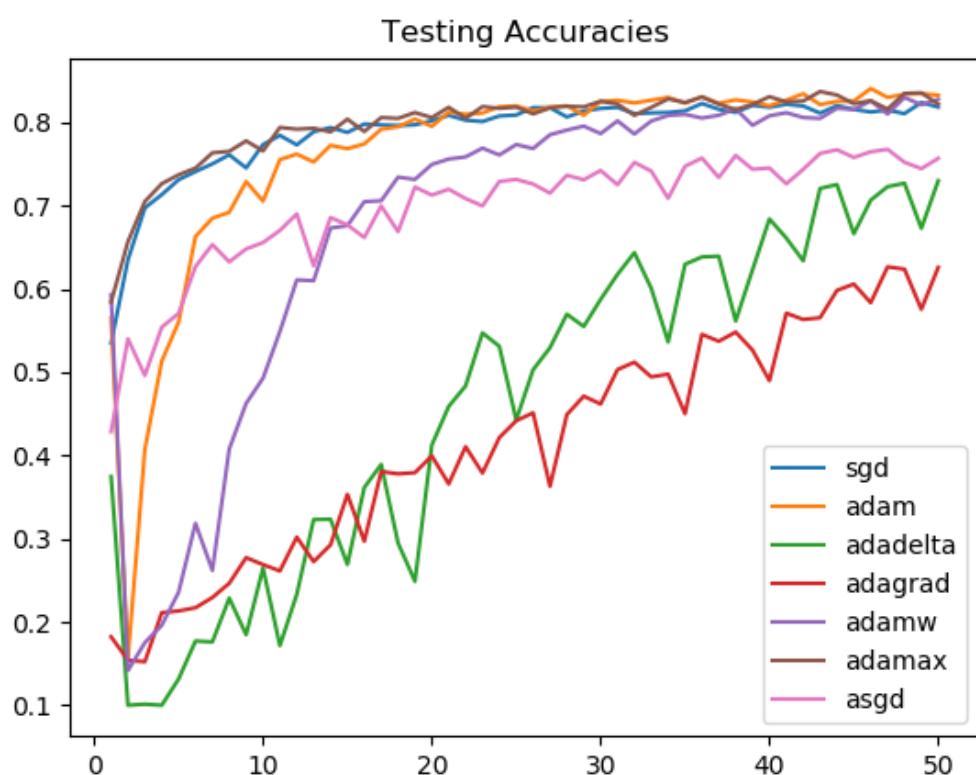


Figura 3.7: Accuratezza: confronto tra i due ottimizzatori Adam ed SGD

Capitolo 4

Esperimenti

Durante la fase sperimentale dello studio, lo strumento utilizzato per condurre gli esperimenti è stato *Colaboratory*, un prodotto di *Google Research* che consente a chiunque, nella sua versione base e gratuita, di scrivere ed eseguire codice *Python* tramite browser ed è particolarmente adatto per l'apprendimento automatico e l'analisi dei dati. Più nello specifico, *Colab*, è un servizio di *Jupyter notebook* che non richiede nessuna configurazione per l'uso, fornendo al contempo l'accesso gratuito a risorse compresa la GPU. Tuttavia, l'accesso alle risorse non è garantito e l'utilizzo delle stesse non è illimitato. Ciò è indispensabile affinché *Colab* possa fornire risorse gratuitamente. [36]

4.1 Analisi dei dataset utilizzati: CASIA v2.0 vs NC16

I *dataset* di riferimento, utilizzati nella fase relativa all'esecuzione degli esperimenti sulla base del modello descritto in precedenza, sono il CASIA V2.0 e l'NC16. Questi due tipi di *dataset* si differenziano non solo per numero di immagini contenute, ma anche per difficoltà di riconoscimento delle alterazioni. Infatti, com'è possibile notare in figura 4.1, l'NC16 presenta, all'interno delle sue immagini, modifiche effettuate in modo tale da essere quasi impercettibili all'occhio umano. Contiene 1,124 immagini in totale, di cui un 50% autentiche e un 50% alterate. Il CASIA v2.0 d'altra parte, contiene 12,622 immagini, aventi differenti formati (JPEG, BMP, TIFF) e dimensione (da 240x160, a 900x600), suddivise in 7,491 autentiche (60%) e 5,123 manipolate (40%) attraverso una delle tecniche descritte in precedenza quali *copy-move*, *splicing* e *removal*. Esempi di campioni prelevati dal *dataset* CASIA V2.0 sono mostrati in figura 4.2. Dalle immagini riportate è chiara la differenza tra i due tipi di *dataset*. Infatti, come è possibile vedere in figura 4.2 (b), i bordi sono molto meno regolari rispetto alla figura 4.1. Inoltre, risulta evidente come si tratti di un'immagine che abbia indubbiamente subito un'operazione di *splicing*, lo si nota dall'incoerenza in termini di luminosità e di sgranatura della porzione di immagine riportata.



Figura 4.1: Campione modificato appartenente al dataset NC16



(a) Autentica

(b) Modificata

Figura 4.2: Campioni appartenenti al dataset CASIA V2.0

4.2 Sperimentazione: motivazioni

Introdotti i concetti fondamentali su cui si è basata la fase sperimentale dello studio, è bene effettuare alcune precisazioni. In particolare, per la realizzazione degli esperimenti vengono prelevate 562 immagini autentiche e 562 immagini modificate dal *dataset* CASIA V2.0. Per l'NC16, avendo un numero complessivo di campioni inferiore rispetto a quello precedentemente citato, vengono invece prelevate tutti. In ogni caso, per permettere al modello di essere allenato con un numero sufficiente di immagini, viene adottata la metodologia conosciuta con il nome *data augmentation* per entrambi i *dataset* citati. La scelta di un numero ridotto di immagini è giustificata dalla mancanza di risorse *hardware* indispensabili per minimizzare i tempi di allenamento, altrimenti molto lunghi. Inoltre, si è voluto testare il modello avendo

a disposizione lo stesso numero di campioni per i due *dataset*. Quanto alla metodologia di paragone per valutarne le prestazioni, viene adottata l'accuratezza. Quest'ultima può essere espressa anche attraverso l'utilizzo della *confusion matrix*, suddivisa in quattro campi: True Negative (TN), True Positive (TP), False Negative (FN), False Positive (FP). Dove per TP si intende i campioni autentici e classificati correttamente come tali, TN indica i campioni che hanno subito modifiche e classificati correttamente come modificati, FN indica i campioni autentici ma classificati come modificati e, allo stesso modo, FP indica i campioni modificati e classificati invece come autentici. Un esempio di *confusion matrix* così organizzata viene mostrato in figura 4.3. [37].

		Predicted Class	
		P	N
		P	True Negatives (TN)
Actual Class	P	True Positives (TP)	False Negatives (FN)
	N	False Positives (FP)	True Negatives (TN)

Figura 4.3: Esempio di organizzazione di una *confusion matrix*

4.3 Pipeline del sistema

L'esecuzione di ogni esperimento consta di quattro fasi principali. Di seguito viene riportata la *pipeline* del sistema considerato e descritto in precedenza.

1. Estrazione delle *patch*, come descritto nella sottosezione 3.1.2 del Capitolo 3.
2. Addestramento della CNN con le *patch* estratte, contenenti sia regioni manomesse che regioni integre delle immagini corrispondenti.
3. Estrazione delle funzionalità da immagini mai viste dalla rete, suddivise in *patch* e applicando la fusione delle *feature* dopo l'ultimo strato Convoluzionale della Rete.
4. Utilizzo dell' SVM sulle 400 *feature* estratte nello step precedente per attuare la classificazione finale. [21]

La *pipeline* ad alto livello è mostrata in figura 4.4:

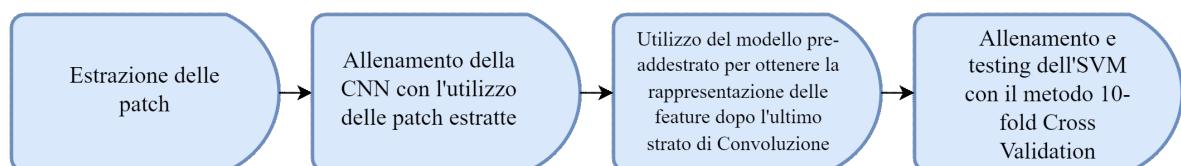


Figura 4.4: Pipeline del sistema

4.4 Fase 1: esperimenti basati sull'utilizzo di SGD

4.4.1 Esperimento 1

Sulla base delle considerazioni fatte precedentemente, la prima fase degli esperimenti si è basata sull'utilizzo del modello descritto nella sezione 3.1 del Capitolo 3. La tabella riportata di seguito mostra i parametri utilizzati per il primo esperimento. In particolare, utilizzando l'ottimizzatore SGD, descritto nella sottosezione 3.1.3 del Capitolo 3.

Tabella 4.1: Esperimento 1: la tabella contiene i parametri utilizzati durante la fase di training

DATASET	LEARNING RATE	NUMERO EPOCHE	BATCH SIZE
CASIA V2.0	0.0001	250	128
NC16	0.0001	250	128

I risultati ottenuti utilizzando i parametri riportati nella tabella precedente sono mostrati in tabella 4.2.

Tabella 4.2: Esperimento 1: la tabella contiene i risultati ottenuti a seguito del training e della classificazione

RISULTATI		
DATASET	TRAINING ACCURACY (%)	VALIDATION ACCURACY (%)
CASIA V2.0	88.3	66.9
NC16	51.4	50.4

La tabella 4.2 mostra come, per il *dataset* CASIA V2.0, in fase di *training* la Rete riesca ad ottenere buoni risultati con l'88.3% di *accuracy* ma, in fase di validazione, rivelano come, in realtà, considerata la netta differenza tra i due valori, il modello abbia scarsa abilità nel generalizzare e quindi, sia poco affidabile. Quanto al *dataset* NC16, i risultati mostrano l'incapacità del modello di classificare correttamente i campioni, risultati confermati anche nelle *confusion matrix* riportate nelle tabelle 4.3 e 4.4.

Tabella 4.3: Esperimento 1: *confusion matrix* contenente i campioni correttamente classificati e quelli classificati in maniera erronea per il *dataset* CASIA V2.0

DATASET CASIA V2.0	PREDICTED POSITIVE	PREDICTED NEGATIVE
ACTUAL POSITIVE	67	45
ACTUAL NEGATIVE	39	74

Tabella 4.4: Esperimento 1: *confusion matrix* contenente i campioni correttamente classificati e quelli classificati in maniera erronea per il *dataset* NC16

DATASET NC16	PREDICTED POSITIVE	PREDICTED NEGATIVE
ACTUAL POSITIVE	113	0
ACTUAL NEGATIVE	112	0

In particolare, per il *dataset* NC16, la tabella 4.4 mostra la netta incapacità del modello di effettuare una corretta classificazione. Infatti, dei 113 campioni negativi, non riesce ad effettuare una corretta classificazione per nessuno di essi. La figura 4.5 riportata di seguito, mostra il *training loss*, ossia la penalità per una cattiva previsione. L'obiettivo dell'addestramento è proprio quello di ridurre al minimo questo valore [38]. In particolare, per il CASIA V2.0 si evince come descrega significativamente dopo le prime epoche e, successivamente, continui a decrescere meno velocemente man mano che le epoche aumentano. Per il *dataset* NC16 si nota, invece, come rimanga stabile senza mai decrescere.

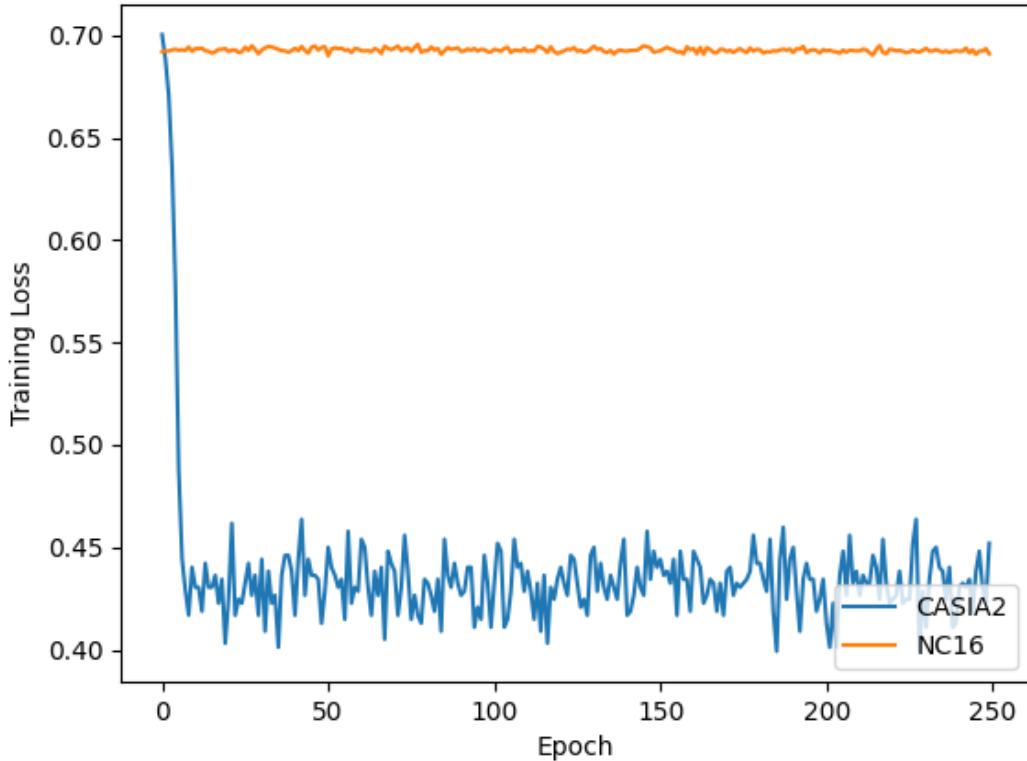


Figura 4.5: *training loss* del primo esperimento calcolato per i due *dataset*

4.4.2 Esperimento 2

Per il secondo esperimento, si è deciso di allenare la Rete modificando il valore del *learning rate* e il numero di epoche. Questo perché, come riportato nella sottosezione 3.1.3 del Capitolo 3 e ampiamente dimostrato in letteratura, il *learning rate* rappresenta uno dei possibili parametri da modificare per valutare come l'utilizzo di un valore maggiore o, al contrario, un valore minore, possa apportare migliorie o, nel peggiore dei casi, decrementare le prestazioni del modello. Quanto alla differenza del numero di epoche utilizzate nel *training* durante l'esecuzione del secondo esperimento, è giustificata dal fatto che, per il *dataset* CASIA V2.0, il *training loss* decresce significativamente dopo le prime epoche, mentre per il secondo non risulta subire lo stesso decremento e dunque, nell'esperimento, si è voluto provare ad aumentarne il numero per valutare l'incremento o meno dell'accuratezza. La tabella 4.5 sintetizza i parametri utilizzati per il secondo esperimento.

Tabella 4.5: Esperimento 2: la tabella contiene i parametri utilizzati durante la fase di training

DATASET	LEARNING RATE	NUMERO EPOCHE	BATCH SIZE
CASIA V2.0	0.001	250	128
NC16	0.001	300	128

I risultati mostrati in tabella 4.6 mostrano come, modificando il *learning rate*, si sia verificato un lieve miglioramento per quanto riguarda la *validation accuracy* per il *dataset* CASIA V2.0 (+1%). I risultati migliori, tuttavia, sono stati ottenuti per l'NC16. Infatti, l'aumento dell'*accuracy* in termini di accuratezza del modello, è pari al +24,8%.

Tabella 4.6: Esperimento 2: la tabella contiene i risultati ottenuti a seguito del training e della classificazione

RISULTATI		
DATASET	TRAINING ACCURACY (%)	VALIDATION ACCURACY (%)
CASIA V2.0	88.3	67.9
NC16	78.5	75.2

Risultati confermati anche dalle *confusion matrix* riportate nelle tabelle 4.7 e 4.8 che mostrano come, per il CASIA V2.0, ci sia un lieve miglioramento nella classificazione dei campioni alterati e, per l'NC16, la corretta classificazione di 84 campioni alterati e 82 campioni originali. Inoltre, la tendenza del modello per il *dataset* NC16, è quella di ottenere un maggior numero, seppur minimo, di falsi negativi (31) rispetto ai falsi positivi (28).

Tabella 4.7: Esperimento 2: *confusion matrix* contenente i campioni correttamente classificati e quelli classificati in maniera erronea per il *dataset* CASIA V2.0

DATASET CASIA V2.0	PREDICTED POSITIVE	PREDICTED NEGATIVE
ACTUAL POSITIVE	71	41
ACTUAL NEGATIVE	41	72

Tabella 4.8: Esperimento 2: *confusion matrix* contenente i campioni correttamente classificati e quelli classificati in maniera erronea per il *dataset* NC16

DATASET NC16	PREDICTED POSITIVE	PREDICTED NEGATIVE
ACTUAL POSITIVE	82	31
ACTUAL NEGATIVE	28	84

In figura 4.6 è riportato il *training loss* del secondo esperimento. Dal grafico si evince che, rispetto a quanto ipotizzato, l'aumentare delle epoche non comporta un significativo decremento del *training loss*, il quale raggiunge il suo valore minimo entro le prime 250 epoche. Per questo motivo, nei test successivi, il numero di epoche sarà riportato a 250.

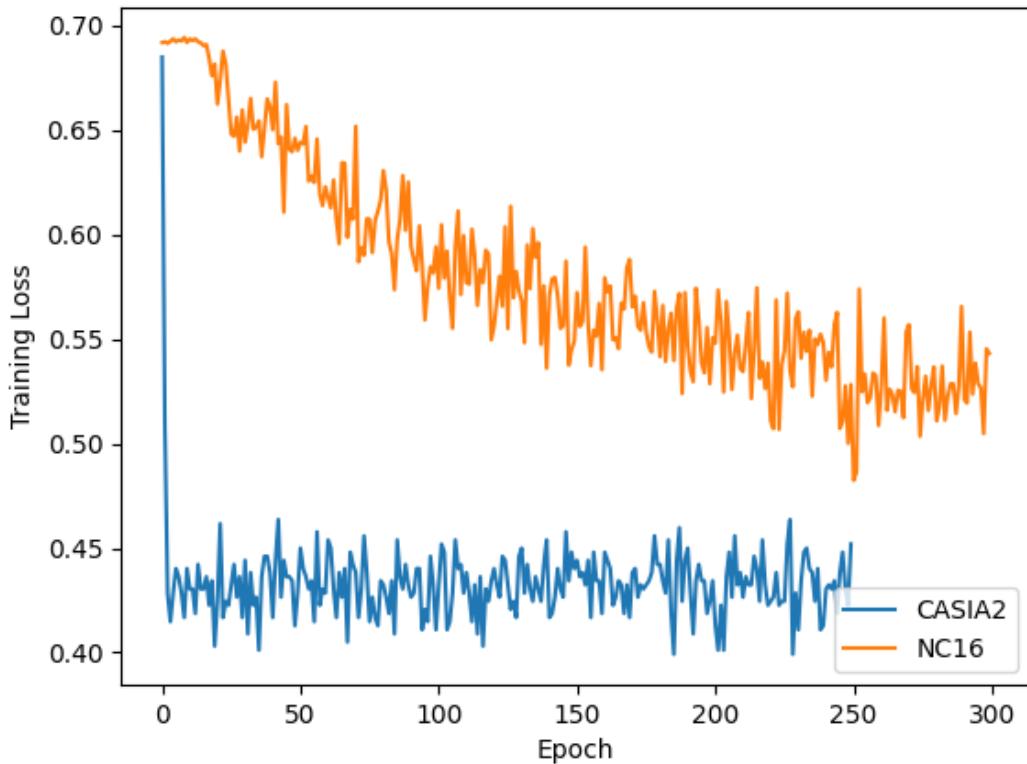


Figura 4.6: *training loss* del secondo esperimento calcolato per i due *dataset*

4.4.3 Esperimento 3

Come metro di paragone aggiuntivo, si è poi provveduto ad eseguire un ulteriore test. In particolare, l’obiettivo dell’esperimento proposto è quello di allenare la CNN su un determinato *dataset* così da testarlo sull’altro e viceversa. Questo tipo di test consente di verificare le prestazioni in ambito di generalizzazione della Rete. Per essere più specifici, si provvede ad effettuare l’estrazione delle *feature* per l’NC16 con il modello preallenato tramite i campioni appartenenti al CASIA V2.0 e, successivamente, si provvede ad effettuare la classificazione tramite SVM. [21] I risultati di questa prova vengono mostrati nella tabella 4.9 e riportata di seguito:

Tabella 4.9: Esperimento 3: accuratezza del modello in termini di generalizzazione

RISULTATI		
DATASET PER IL TRAINING	DATASET PER L'ESTRAZIONE DELLE FEATURE	VALIDATION ACCURACY (%)
CASIA V2.0	NC16	73.8
NC16	CASIA V2.0	65.2

4.5 Fase 2: esperimenti basati sull'utilizzo di Adam

4.5.1 Esperimento 1

Nella seconda fase della sperimentazione si è testato il modello utilizzando, come tecnica di ottimizzazione, l'ottimizzatore Adam. Come per gli esperimenti precedenti, in tabella 4.10, vengono riportati i parametri utilizzati per il primo esperimento della seconda fase.

Tabella 4.10: Esperimento 1: la tabella contiene i parametri utilizzati durante la fase di training

DATASET	LEARNING RATE	NUMERO EPOCHE	BATCH SIZE
CASIA V2.0	0.001	250	128
NC16	0.001	250	128

Quanto ai risultati mostrati in tabella 4.11, illustrano come vi sia stato un miglioramento della *validation accuracy* per il *dataset* NC16 pari allo +0.8%, mentre per quanto riguarda il dataset CASIA V2.0 vi è un decremento pari a -5.7%.

Tabella 4.11: Esperimento 1: la tabella contiene i risultati ottenuti a seguito del training e della classificazione

RISULTATI		
DATASET	TRAINING ACCURACY (%)	VALIDATION ACCURACY (%)
CASIA V2.0	88.4	62.2
NC16	84.2	76.0

Per meglio comprendere l'esito di tale prova, tramite le tabelle 4.12 e 4.13, vengono riportate anche le *confusion matrix* relative alla classificazione dei campioni effettuata dal modello. In questo caso la tendenza del modello è quella di ottenere, per entrambi i *dataset*, un maggior numero di falsi positivi.

Tabella 4.12: Esperimento 1: *confusion matrix* contenente i campioni correttamente classificati e quelli classificati in maniera erronea per il *dataset* CASIA V2.0

DATASET CASIA V2.0	PREDICTED POSITIVE	PREDICTED NEGATIVE
ACTUAL POSITIVE	78	34
ACTUAL NEGATIVE	54	59

Tabella 4.13: Esperimento 1: *confusion matrix* contenente i campioni correttamente classificati e quelli classificati in maniera erronea per il *dataset* NC16

DATASET NC16	PREDICTED POSITIVE	PREDICTED NEGATIVE
ACTUAL POSITIVE	85	28
ACTUAL NEGATIVE	34	78

4.5.2 Esperimento 2

Considerato il miglioramento ottenuto per il *dataset* NC16, si è ritenuto opportuno effettuare la riesecuzione della prova per valutarne un eventuale miglioramento. I parametri utilizzati, essendo una riesecuzione, sono rimasti invariati come riportato nella tabella 4.14.

Tabella 4.14: Esperimento 2: la tabella contiene i parametri utilizzati durante la fase di training

DATASET	LEARNING RATE	NUMERO EPOCHE	BATCH SIZE
NC16	0.001	250	128

La tabella 4.15 riportata di seguito, mostra come la riesecuzione abbia favorito un ulteriore, seppur minimo, aumento della *validation accuracy* dello +0.2%. In totale quindi, l'aumento dell'accuratezza del modello per il dataset NC16 è pari al +1.0%.

Tabella 4.15: Esperimento 2: la tabella contiene i risultati ottenuti a seguito del training e della classificazione

RISULTATI		
DATASET	TRAINING ACCURACY (%)	VALIDATION ACCURACY (%)
NC16	85.5	76.2

La *confusion matrix* riportata in tabella 4.16, mostra la tendenza del modello, in questo caso, di ottenere più falsi positivi (30) che falsi negativi (10). La Rete risulta comunque essere in grado di classificare correttamente 103 campioni originali e 82 contenenti alterazioni.

Tabella 4.16: Esperimento 2: *confusion matrix* contenente i campioni correttamente classificati e quelli classificati in maniera erronea per il *dataset* NC16

DATASET NC16	PREDICTED POSITIVE	PREDICTED NEGATIVE
ACTUAL POSITIVE	103	10
ACTUAL NEGATIVE	30	82

4.5.3 Esperimento 3

Infine, come ultimo esperimento, si è deciso di valutare il grado di generalizzazione del modello rispetto alle modifiche apportate e l'utilizzo dell'ottimizzatore Adam. I risultati ottenuti, mostrano come ci sia stato un miglioramento per quanto riguarda l'*accuracy* (+1.1%) utilizzando il dataset NC16 per allenare la rete e che, in generale, la CNN risulta avere prestazioni migliori per il dataset NC16 rispetto al CASIA V2.0. Di seguito viene riportata la tabella riassuntiva contenente i valori ottenuti:

Tabella 4.17: Esperimento 3: generalizzazione - confronto risultati tra i due ottimizzatori Adam ed SGD

RISULTATI			
DATASET PER IL TRAINING	DATASET PER L'ESTRAZIONE DELLE FEATURE	VALIDATION ACCURACY (%)	OTTIMIZZATORE
CASIA V2.0	NC16	73.8	SGD
NC16	CASIA V2.0	65.2	SGD
CASIA V2.0	NC16	68.6	ADAM
NC16	CASIA V2.0	66.3	ADAM

È possibile effettuare una riflessione sulla base del numero differente di campioni utilizzati per l'esecuzione delle prove condotte. In particolare, è interessante effettuare un parallelismo rispetto ai risultati ottenuti utilizzando la totalità dei campioni appartenenti al *dataset* CASIA V2.0 [21]. Nelle ultime due righe della tabella 4.18, infatti, sono riportati i valori ottenuti utilizzando solo 1.124 immagini del *dataset* citato mentre nelle prime due, utilizzando il *dataset* nella sua interezza. Dunque, l'impiego di un numero inferiore di immagini e l'utilizzo di un altro tipo di ottimizzatore, potrebbe spiegare i risultati differenti ottenuti in termini di prestazioni.

Tabella 4.18: Risultati, in termini di accuratezza della classificazione, degli esperimenti effettuati con il *dataset* CASIA V2.0 completo e gli esperimenti, fulcro dell'elaborato e del lavoro di tesi proposto, utilizzando il *dataset* CASIA 2.0 incompleto

RISULTATI			
DATASET PER IL TRAINING	DATASET PER L'ESTRAZIONE DELLE FEATURE	VALIDATION ACCURACY (%)	
CASIA V2.0	NC16	67.5	
NC16	CASIA V2.0	59.8	
CASIA V2.0	NC16	68.6	
NC16	CASIA V2.0	66.3	

4.6 Considerazioni e risultati finali

Una volta riportati gli esiti delle prove in termini percentuali, è possibile effettuare una considerazione basata sull’ispezione visiva dei campioni classificati in maniera errata dal modello. A titolo esemplificativo, di seguito, vengono riportate delle immagini rappresentanti i campioni non classificati correttamente, appartenenti ai due *dataset*. È possibile notare come i campioni classificati in maniera errata presentino delle similitudini. In particolare, sono presenti elementi fuori fuoco, come in figura 4.7 (a), riflessi su superfici non regolari, nel caso dell’immagine in figura 4.7 (b) e, ancora, elementi poco nitidi. Queste osservazioni ci permettono di supporre che il modello, in presenza di questi elementi, abbia difficoltà nell’eseguire una corretta classificazione. Ciò è chiaramente visibile in figura 4.7 (b), rappresentante un campione originale appartenente al *dataset* NC16 ed erroneamente classificato come modificato. Al contrario, in figura 4.7 (a), il campione appartenente alla categoria di immagini modificate, viene classificato erroneamente come autentico.



(a) Campione appartenente al *dataset* CASIA



(b) Campione appartenente al *dataset* NC16

Figura 4.7: Esempio immagini classificate erroneamente dal modello.

Capitolo 5

Conclusioni

Nello studio effettuato si è voluto sperimentare l'utilizzo di una Rete Neurale Convoluzionale come strumento per affrontare il problema dell'*Image Forgery Detection* e, quindi, come mezzo di difesa al problema più generale della diffusione di *fake news*. Per farlo, sono stati adoperati due *dataset* differenti, sia in termini numerici che per tipologia dei campioni presenti al loro interno. Ricordiamo, infatti, che il *dataset* NC16 è un dataset organizzato in modo tale da risultare molto più difficile, rispetto al CASIA V2.0, riconoscere ad occhio nudo i campioni manomessi. L'obiettivo dello studio proposto è stato dunque valutare le prestazioni della Rete, considerate le differenze tra i due *dataset*, e determinare come l'utilizzo di un differente ottimizzatore potesse influenzarne il comportamento. I risultati raggiunti tramite l'utilizzo dell'ottimizzatore Adam, 76.2% e 62.2% rispettivamente per il *dataset* NC16 e CASIA V2.0, mostrano come, differentemente da quanto ipotizzato inizialmente, per il primo si siano ottenute prestazioni migliori. La validità di questa osservazione però, è limitata all'utilizzo dello stesso numero di campioni prelevati dai due *dataset*. Altra importante riflessione riguarda l'uso degli iperparametri e come una piccola variazione del valore del *learning rate* abbia influenzato enormemente le prestazioni del modello. Si è passati infatti, modificando il valore da 0.0001 a 0.001, da una *validation accuracy* del 50.4% ad una *validation accuracy* del 75.2% per il dataset NC16. Quanto al CASIA V2.0, non vi è stato lo stesso incremento. Si è passati infatti, da una *validation accuracy* del 66.9% ad una del 67.9%. Per quanto concerne l'utilizzo dell'ottimizzatore Adam, il suo impiego ha permesso di ottenere un ulteriore incremento dell'accuratezza per il *dataset* NC16, passando da un'*accuracy* del 75.2% ad un'*accuracy* del 76.2%. Lo stesso si è mostrato meno adatto per il *dataset* CASIA V2.0 per il quale si è verificato invece un decremento, si è passati infatti da un'accuratezza del 67.9% ad un'accuratezza del 62.2%. La motivazione, come già accennato in precedenza, potrebbe essere legata all'utilizzo parziale del numero di campioni disponibili per il *dataset* citato. Dunque, dalle considerazioni effettuate, come studio futuro si potrebbe pensare di ripetere la sperimentazione utilizzando il dataset CASIA V2.0 nella sua interezza. Altro spunto interessante, considerate le valutazioni proposte all'inizio e riguardanti la possibilità di favorire la diffusione di *fake news* soprattutto attraverso i social, potrebbe essere quello di testare il modello su un *dataset* contenente immagini prelevate da questi ultimi. Quanto detto fino ad ora viene riportato nella tabella 5.1, contenente la sintesi delle evidenze ottenute con l'esecuzione degli

esperimenti riportati nel Capitolo 4.

Tabella 5.1: La tabella mostra la sintesi di tutti i risultati ottenuti durante la sperimentazione sia tenendo conto del valore del *learning rate* utilizzato, che considerando i differenti ottimizzatori impiegati

RISULTATI				
DATASET PER IL TRAINING	DATASET PER L'ESTRAZIONE DELLE FEATURE	LEARNING RATE	OTTIMIZZATORE	VALIDATION ACCURACY (%)
CASIA V2.0	CASIA V2.0	0.0001	SGD	66.9
NC16	NC16	0.0001	SGD	50.4
CASIA V2.0	CASIA V2.0	0.001	SGD	67.9
NC16	NC16	0.001	SGD	75.2
CASIA V2.0	NC16	0.001	SGD	73.8
NC16	CASIA V2.0	0.001	SGD	65.2
CASIA V2.0	CASIA V2.0	0.001	ADAM	62.2
NC16	NC16	0.001	ADAM	76.2
CASIA V2.0	NC16	0.001	ADAM	68.6
NC16	CASIA V2.0	0.001	ADAM	66.3

Bibliografia

- [1] In: (). URL: <https://www.history.com/news/josef-stalin-great-purge-photo-retouching>.
- [2] In: (). URL: <https://www.businessinsider.com/fake-photos-history-2011-8?r=US&IR=T#the-cottingley-fairies-were-actually-paper-cutouts-created-by-two-young-girls-between-1917-1920-10>.
- [3] In: (). URL: <https://blog.paperspace.com/deep-learning-metrics-precision-recall-accuracy/>.
- [4] “Digital (Consumer) Evolution”. In: (2021), pp. 13–15.
- [5] Sebastiano Battiato, Giuseppe Messina e Rosetta Rizzo. “Image Forensics Contraffazione Digitale e Identificazione della Camera di Acquisizione: Status e Prospettive”. In: (feb. 2022).
- [6] J.A. Redi et al. “Digital image forensics: a booklet for beginners”. In: *Multimedia Tools and Applications* 51 (2011), pp. 133–162. DOI: [10.1007/s11042-010-0620-1](https://doi.org/10.1007/s11042-010-0620-1).
- [7] Erwin Quiring, Daniel Arp e Konrad Rieck. “Fraternal Twins: Unifying Attacks on Machine Learning and Digital Watermarking”. In: (mar. 2017), p. 3.
- [8] Akram Hatem Saber, Mohd Ayyub Khan e Basim Galeb Mejbel. “A Survey on Image Forgery Detection Using Different Forensic Approaches”. In: *Advances in Science, Technology and Engineering Systems Journal* 5.3 (2020), pp. 361–370. DOI: [10.25046/aj050347](https://doi.org/10.25046/aj050347).
- [9] Yuecong Lai et al. “An improved block-based matching algorithm of copy-move forgery detection”. In: (2018). DOI: [10.1007/s11042-017-5094-y](https://doi.org/10.1007/s11042-017-5094-y).
- [10] Ishan Jain e Nidhi Goel. “Advancements in Image Splicing and Copy-move Forgery Detection Techniques: A Survey”. In: *2021 11th International Conference on Cloud Computing, Data Science Engineering (Confluence)*. 2021, pp. 470–475. DOI: [10.1109/Confluence51648.2021.9377104](https://doi.org/10.1109/Confluence51648.2021.9377104).
- [11] Kamruzzaman J. Islam M.M. et al. “Passive Detection of Splicing and Copy-Move Attacks in Image Forgery”. In: *Neural Information Processing* 11304 (2018), pp. 555–567. DOI: [10.1007/978-3-030-04212-7_49](https://doi.org/10.1007/978-3-030-04212-7_49).
- [12] Güzin Ulutas e Gul Muzaffer. “A New Copy Move Forgery Detection Method Resistant to Object Removal with Uniform Background Forgery”. In: *Mathematical Problems in Engineering* 2016 (2016), pp. 1–19.

- [13] Saad Albawi, Tareq Abed Mohammed e Saad Al-Zawi. "Understanding of a convolutional neural network". In: *2017 International Conference on Engineering and Technology (ICET)*. 2017, pp. 1–6. DOI: [10.1109/ICEngTechnol.2017.8308186](https://doi.org/10.1109/ICEngTechnol.2017.8308186).
- [14] In: (). URL: <https://www.seldon.io/transfer-learning/>.
- [15] In: (). URL: <https://www.ai4business.it/intelligenza-artificiale/transfer-learning-cose-come-funziona-e-applicazioni/>.
- [16] Yash Shah et al. "Deep Learning model-based Multimedia forgery detection". In: *2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*. 2020, pp. 564–572. DOI: [10.1109/I-SMAC49090.2020.9243530](https://doi.org/10.1109/I-SMAC49090.2020.9243530).
- [17] Anushka Singh e Jyotsna Singh. "Image forgery detection using Deep Neural Network". In: *2021 8th International Conference on Signal Processing and Integrated Networks (SPIN)*. 2021, pp. 504–509. DOI: [10.1109/SPIN52536.2021.9565953](https://doi.org/10.1109/SPIN52536.2021.9565953).
- [18] In: (). URL: <https://medium.com/@mygreatlearning/what-is-vgg16-introduction-to-vgg16-f2d63849f615>.
- [19] In: (). URL: <https://viso.ai/deep-learning/resnet-residual-neural-network/>.
- [20] Yuan Rao e Jiangqun Ni. "A deep learning approach to detection of splicing and copy-move forgeries in images". In: (2016), pp. 1–6. DOI: [10.1109/WIFS.2016.7823911](https://doi.org/10.1109/WIFS.2016.7823911).
- [21] In: (). URL: <https://github.com/kPsarakis/Image-Forgery-Detection-CNN>.
- [22] In: (). URL: <https://prateekvjoshi.com/2016/04/05/what-is-local-response-normalization-in-convolutional-neural-networks/>.
- [23] In: (). URL: <https://towardsdatascience.com/dont-overfit-ii-how-to-avoid-overfitting-in-your-machine-learning-and-deep-learning-models-2ff903f4b36a>.
- [24] In: (). URL: <https://www.ibm.com/cloud/learn/overfitting#toc-how-to-avoid-ODt1>.
- [25] In: (). URL: <https://deepai.org/machine-learning-glossary-and-terms/softmax-layer>.
- [26] In: (). URL: <https://www.geeksforgeeks.org/dropout-in-neural-networks/>.
- [27] Esperanza García-Gonzalo et al. "Hard-Rock Stability Analysis for Span Design in Entry-Type Excavations with Learning Classifiers". In: *Materials* 9 (giu. 2016), p. 531. DOI: [10.3390/ma9070531](https://doi.org/10.3390/ma9070531).
- [28] In: (). URL: <https://www.eage.it/machine-learning/support-vector-machine>.
- [29] In: (). URL: <https://machinelearningmastery.com/k-fold-cross-validation/>.
- [30] In: (). URL: <https://peltarion.com/knowledge-center/documentation/modeling-view/run-a-model/optimizers/stochastic-gradient-descent>.
- [31] In: (). URL: <https://scikit-learn.org/stable/modules/sgd.html>.
- [32] In: (). URL: <https://towardsdatascience.com/adam-latest-trends-in-deep-learning-optimization-6be9a291375c>.

- [33] In: (). URL: <https://towardsdatascience.com/deep-learning-optimizers-436171c9e23f>.
- [34] In: (). URL: <https://medium.com/geekculture/a-2021-guide-to-improving-cnns-optimizers-adam-vs-sgd-495848ac6008>.
- [35] In: (). URL: <https://blog.paperspace.com/optimization-in-deep-learning/>.
- [36] In: (). URL: <https://research.google.com/colaboratory/faq.html>.
- [37] Karel Horak. “Classification of SURF Image Features by Selected Machine Learning Algorithms”. In: (lug. 2017).
- [38] In: (). URL: <https://developers.google.com/machine-learning/crash-course/descending-into-ml/training-and-loss>.

Ringraziamenti

Rivolgo un ringraziamento speciale al Professore Andrea Francesco Abate e al Professore Fabio Narducci per la disponibilità e il supporto concessomi; per avermi dato questa importante opportunità di crescita, cruciale per la realizzazione di questo traguardo. Ringrazio i miei genitori e mio fratello per essere stati un punto di riferimento fondamentale durante questi anni; per la vicinanza e l'affetto con cui mi sono sempre stati accanto nei momenti felici e soprattutto in quelli di sconforto; per aver sempre creduto in me.