



UNIVERSITÀ
DI TRENTO

Dipartimento di Ingegneria e Scienza dell' Informazione

Corso di Laurea in
Ingegneria Informatica, delle Comunicazioni ed Elettronica

ELABORATO FINALE

STUDIO SULLA RILEVAZIONE EFFICIENTE DI
DEEPFAKES ATTRaverso
APPRENDIMENTO AUTO SUPERVISIONATO

Supervisore

Prof. Giulia Boato

Laureando

Lorenzo Marogna

Correlatori

Andrea Montibeller, PhD

Francesco Paissan

Elisabetta Farella, PhD

Anno Accademico 2023/2024

Ringraziamenti

Al termine di questo lavoro, ritengo essenziale dedicare alcune righe a coloro che, senza la minima esitazione, hanno contribuito alla realizzazione di questo progetto. In primis, è mio dovere ringraziare la mia relatrice, Giulia Boato, e il mio correlatore, Andrea Montibeller, per la loro pazienza e disponibilità, e per avermi accompagnato durante questa importante fase del mio percorso accademico. Un sentito ringraziamento va alla Fondazione Bruno Kessler per avermi dato l'opportunità di sperimentare in prima persona una realtà di ricerca grande e importante, permettendomi di sviluppare la curiosità e l'interesse che successivamente mi hanno portato alla creazione di questo lavoro. Ringrazio inoltre i miei tutor, Francesco Paissan e la Dott.ssa Elisabetta Farella, insieme a tutti i colleghi dell'ufficio E3DA, per i preziosi consigli e per avermi guidato all'interno della realtà FBK. Un solo grazie non basterebbe per ripagare la mia famiglia di tutto l'affetto e il sostegno ricevuto durante il mio percorso di vita: grazie a papà e mamma per avermi permesso di arrivare fin qui e per aver creduto in me fin dall'inizio. Infine, ringrazio tutti gli amici e i compagni di corso che, vivendo con me questa avventura, hanno dato un ulteriore significato al mio percorso accademico.

Indice

Sommario	2
1 Lavori Correlati	5
1.1 Conoscenze di base	5
1.1.1 Paradigmi di Apprendimento	5
1.1.2 Generazione di Deepfakes	6
1.2 Rilevazione di Deepfakes	9
1.3 Architetture per dispositivi a risorse limitate	11
1.3.1 Limitazioni su Micro Controllori	11
1.3.2 PhiNets	11
1.3.3 MicroNet	13
2 Metodi	15
2.1 Apprendimento Supervisionato Pre-Social e Fine Tuning Post-Social	15
2.2 Apprendimento Auto-Supervisionato con Fine Tuning Supervisionato	17
2.2.1 SimCLR: Un Framework di Apprendimento Contrastivo	17
3 Setup Sperimentale	22
3.1 Dataset	22
3.2 Architetture	23
3.3 Apprendimento Supervisionato	23
3.4 Apprendimento Auto-Supervisionato	23
3.5 Metriche	24
4 Risultati	26
Conclusioni	29
Bibliografia	30

Sommario

Il concetto di *Deepfake*, emerso di recente, si riferisce comunemente a contenuti digitali fintizi, creati alterando materiale esistente o generando immagini completamente nuove [31]. Il termine 'deep' indica l'uso di algoritmi di Deep Learning (DL), un potente ramo del Machine Learning (ML) con applicazioni nella computer vision, nel riconoscimento vocale e in vari altri campi [38]. Il deep learning ha raggiunto risultati impressionanti, talvolta superando persino gli esseri umani. Mentre un tempo i deepfake erano più facili da individuare, i progressi nell'intelligenza artificiale (IA) e l'aumento della potenza di calcolo hanno alimentato lo sviluppo di tecniche di generazione di immagini altamente realistiche. Ciò ha portato alla creazione di app di facile utilizzo in grado di generare deepfake in tempo reale. La diffusione di questi contenuti falsi sui social media solleva notevoli preoccupazioni sulla disinformazione e sull'affidabilità delle informazioni online. Individuare i deepfake è ancora più complesso nell'ambito della generazione di volti. Persino osservatori umani addestrati faticano, con studi come [4] che mostrano come l'accuratezza umana vari tra il 30% e l'85% a seconda dell'immagine, con una media intorno al 62%. Inoltre, l'articolo [4] mostra che per un'immagine su cinque l'accuratezza non raggiunge il 50%, rendendo preferibile il lancio di una moneta al giudizio umano. I risultati sono simili in [31], dove le prestazioni umane sono state testate contro tecnologie di anni più e meno recenti. Qui, gli umani hanno etichettato le immagini sintetiche come reali il 68% delle volte, mentre hanno identificato correttamente le immagini reali solo il 52% delle volte. Questi risultati evidenziano l'urgente necessità di strumenti di rilevamento automatizzati, specialmente in aree che si basano sull'autenticità, come la verifica degli utenti e la lotta alla diffusione della disinformazione.

Per affrontare questa sfida, i ricercatori hanno creato modelli di IA che rilevano i deepfake analizzando i pattern e gli artefatti lasciati durante il processo di generazione [20]. Tuttavia, la condivisione sui social media spesso degrada queste tracce comprimendole con algoritmi privati, specifici per ogni piattaforma [1]. Questo processo rende difficile il rilevamento, ostacolando le prestazioni dei modelli a seconda della piattaforma che ha compresso l'immagine. Per combattere questo problema in contesti reali sui social media, è stato creato un dataset chiamato 'TrueFace' [3]. Contiene immagini reali e false da Facebook, Telegram e Twitter, permettendo ai ricercatori di addestrare modelli su immagini soggette alla compressione dei social media. La ricerca in corso si concentra sullo sviluppo di tecniche che consentano ai modelli di generalizzare su diverse piattaforme, indipendentemente dai processi di condivisione ai quali il contenuto è stato sottoposto.

Machine Learning su dispositivi a risorse limitate L'Internet of Things (IoT) consiste in una vasta rete di dispositivi intelligenti interconnessi. Per elaborare efficacemente l'enorme quantità di dati che producono, dobbiamo spostare l'intelligenza dai server cloud centralizzati agli estremi della rete, più vicino ai dispositivi e ai sensori. Tuttavia, l'elaborazione fatta ai margini presenta delle difficoltà a causa di vincoli di risorse come memoria limitata, utilizzo di banda di comunicazione e potenza della batteria, come mostrato in [1]. Queste limitazioni rendono difficile portare algoritmi di AI tradizionali su dispositivi periferici come i sensori. Per affrontare queste problematiche e portare le capacità dell'AI agli estremi della rete, è emerso il Tiny Machine Learning (TinyML) [24]. Il TinyML si concentra sullo sviluppo di modelli e algoritmi di machine learning efficienti che possono operare entro i vincoli dei dispositivi edge. Questa tecnologia offre numerosi vantaggi nell'elaborazione dei dati dei sensori su dispositivi integrati a basso consumo. Poiché non è richiesta connettività per l'inferenza, i dati possono essere elaborati in tempo reale direttamente sul dispositivo, evitando la necessità di doverli trasferire grezzi a terze parti per l'elaborazione. Ciò elimina i ritardi dovuti all'attesa di tempi di trasmissione non necessari e riduce significativamente la latenza. Per comprendere l'impatto della bassa latenza di risposta, si consideri il monitoraggio continuo delle macchine in ambienti industriali per prevedere problemi e potenziali guasti. Questo tipo di applicazione può fornire una risposta tempestiva ai danni, riducendo i costi di manutenzione, i rischi di guasto e i tempi di inattività, migliorando al contempo

le prestazioni e l'efficienza. I dispositivi integrati che supportano gli algoritmi TinyML richiedono una quantità molto piccola di energia (nell'ordine dei milliwatt), il che permette loro di funzionare per lunghi periodi senza bisogno di essere ricaricati se dotati di batteria (vedi esempio in [40]), o con un consumo energetico minimo se alimentati. La ricerca per ridurre i requisiti energetici non mira solo a portare l'AI sui microcontrollori. L'addestramento di grandi modelli di Machine Learning da parte delle aziende più potenti del mondo come Microsoft, Meta o Google, richiede molta energia per il loro addestramento e implementazione. Attualmente, gli investitori sostengono la ricerca in questa direzione, e solo le grandi aziende possono gestire l'onere finanziario. Ma poiché i modelli richiedono risorse sempre maggiori, la sostenibilità a lungo termine di questo approccio rimane incerta. Studi recenti hanno cercato di valutare le emissioni di CO₂ di grandi modelli linguistici come BLOOM o GPT-3 [42, 34] mostrando risultati preoccupanti. Inoltre, la competizione tra le grandi aziende tecnologiche per il miglior modello sta aggravando la situazione. Ad esempio, ChatGPT di OpenAI e Gemini di Google sono entrambi computazionalmente costosi e potenzialmente contribuiscono a impatti ambientali simili, nonostante soddisfino scopi simili. Questa ridondanza potrebbe portare a un aumento del consumo energetico senza significativi progressi nella funzionalità. Spostare il rilevamento dei deepfake sui dispositivi edge offre diversi vantaggi. In primo luogo, espande le capacità dei microcontrollori (MCU) con un nuovo compito in un ambito insolito per questo mondo. In secondo luogo, ridurre il numero di operazioni accelera il tempo di inferenza (il tempo necessario al modello per analizzare e giudicare un'immagine). Un'inferenza più rapida ci avvicina alle applicazioni in tempo reale. Sebbene l'analisi frame per frame dei deepfake per i video possa non sembrare immediatamente necessaria, sicuramente non è nociva. Un vantaggio significativo è che requisiti di implementazione inferiori permettono ai dispositivi di eseguire l'inferenza localmente [40]. Ad esempio, una piccola rete su uno smartphone potrebbe verificare l'autenticità dei contenuti senza utilizzare la banda. Più in generale, un'inferenza più rapida è cruciale dato l'enorme numero di immagini caricate online ogni secondo. Statistiche non ufficiali suggeriscono che Instagram vede circa mille caricamenti di immagini al secondo, di cui il 34% coinvolge soggetti umani. Se l'obiettivo è verificare la realtà di ogni immagine prima di pubblicarla, l'efficienza è fondamentale.

Metodi e Risultati Questa tesi affronta la necessità critica di adattare il rilevamento dei deepfake alle sfide del mondo reale. Si concentra su due strategie chiave: migliorare la resilienza del modello contro gli artefatti di compressione dei social media e consentire un efficiente rilevamento dei deepfake su dispositivi a risorse limitate. La prima strategia sfrutta l'Apprendimento Auto-Supervisionato (Self Supervised Learning - SSL) per migliorare la robustezza del modello contro le distorsioni introdotte dagli algoritmi di compressione dei social media. Addestrando i modelli su dati non etichettati per apprendere strutture e relazioni intrinseche, SSL migliora la loro capacità di riconoscere i deepfake nonostante la compressione. Un'esplorazione dettagliata di SSL è fornita nella Sezione 1.1.1. I risultati sperimentali dimostrano che SSL mitiga significativamente il degrado delle prestazioni sulle immagini compresse, riducendo il divario di accuratezza tra contenuti originali e manipolati. Tuttavia, questa maggiore robustezza potrebbe avvenire a scapito dell'accuratezza complessiva di rilevamento, evidenziando la necessità di una attenta considerazione dei compromessi basati sui requisiti specifici dell'applicazione. La seconda strategia indaga architetture di reti neurali leggere, MicroNet [33] e PhiNet [40], per il rilevamento di deepfake su dispositivo. Queste architetture privilegiano un numero ridotto di parametri e una complessità computazionale ridotta, essenziali per l'implementazione su dispositivi a risorse limitate.

I risultati della ricerca mostrano che in condizioni controllate, questi modelli compatti possono raggiungere un'accuracy paragonabile a quella di architetture più grandi come ResNet50 [22]. Ad esempio, l'accuracy di MicroNet su immagini pre-social media è solo marginalmente inferiore a quella di ResNet50, pur utilizzando una frazione dei parametri. Questi risultati sono molto promettenti per consentire l'analisi dei deepfake in tempo reale direttamente sui dispositivi degli utenti, potenzialmente mitigando così la diffusione di contenuti fuorvianti prima che raggiungano un pubblico più ampio.

Struttura dell’ elaborato Questa tesi è organizzata in quattro capitoli. Il Capitolo Uno fornisce informazioni di base sui paradigmi di apprendimento automatico, sulle tecniche di generazione dei deepfake, sui metodi all’avanguardia di rilevamento dei deepfake e sulle architetture edge, incluse Phi-Net [40] e MicroNet [33]. Il Capitolo Due presenta la metodologia di ricerca. Inizia stabilendo una base di riferimento utilizzando un approccio standard di apprendimento automatico per il confronto con i risultati SSL. Successivamente, viene descritta in dettaglio la tecnica SSL, in particolare l’Apprendimento Contrastivo (Contrastive Learning - CL) utilizzando il framework SimCLR. I Capitoli Tre e Quattro forniscono rispettivamente il setup sperimentale, che consente la riproducibilità, e i risultati della ricerca. Questi ultimi sono accompagnati da un’analisi per comprendere come questo lavoro possa far progredire il rilevamento dei deepfake nel mondo reale.

1 Lavori Correlati

Questo capitolo fornisce una panoramica dei deepfake, dalla creazione al rilevamento. Inizialmente, vengono stabilite le conoscenze fondamentali sui principali paradigmi di apprendimento automatico (apprendimento supervisionato non supervisionato e auto-supervisionato) e vengono esplorate alcune tecniche popolari utilizzate nella generazione dei deepfake. Successivamente, vengono esaminati criticamente i metodi più all'avanguardia per il rilevamento dei deepfake, evidenziandone i punti di forza e di debolezza. Il capitolo esplora poi il mondo delle architetture edge, concentrandosi sulle sfide dell'implementazione di modelli TinyML su dispositivi a risorse limitate. Infine, vengono introdotti Phinet e Micronet, le due architetture edge centrali per questo lavoro.

1.1 Conoscenze di base

1.1.1 Paradigmi di Apprendimento

Questa sezione esplorerà alcuni dei principali paradigmi di apprendimento nel campo del machine learning: Apprendimento Supervisionato (Supervised Learning), Apprendimento Non Supervisionato (Unsupervised Learning) e Apprendimento Auto-Supervisionato (Self Supervised Learning).

Il Supervised Learning è un paradigma di apprendimento in cui un algoritmo viene addestrato su un dataset etichettato. Ciò significa che il dataset include sia i dati di input (caratteristiche) che le corrispondenti etichette di output corrette. L'obiettivo è che l'algoritmo apprenda la relazione tra i dati di input e output per fare previsioni accurate su nuovi dati non visti. Questo paradigma si adatta molto bene ai problemi di classificazione come il Filtraggio dello Spam, il Riconoscimento di Immagini e il Rilevamento di Frodi, dove la classe corretta dell'output è rappresentata dall'etichetta. Questa chiara dipendenza tra l'input e l'output atteso di solito si traduce in prestazioni elevate e misurabili per il modello. Tuttavia, i dataset etichettati sono tipicamente più difficili da trovare perché spesso richiedono lavoro umano o esperienza per classificare gli elementi manualmente. Ad esempio, etichettare immagini mediche o classificare transazioni finanziarie complesse richiede specialisti in grado di interpretare accuratamente i dati.

Ne consegue che la ricerca ha sviluppato anche alcune tecniche per sfruttare i dati non etichettati. Con l' Unsupervised Learning, non ci sono output target predefiniti o etichette associate ai dati di input. Invece, l'algoritmo mira a scoprire autonomamente strutture, pattern o relazioni nascoste all'interno dei dati. Questo paradigma può essere utilizzato in molteplici casi d'uso. Ad esempio, identificare diversi gruppi di clienti di un negozio in base ai loro comportamenti di acquisto è un problema di clustering che spesso ha un grande dataset non etichettato. Un'altra classe di problemi affrontati dall'Unsupervised Learning può essere il rilevamento di anomalie, che consiste nell'identificare punti dati outlier da un set più ampio. Per esempio, le reti generative avversarie (GAN) sfruttano dataset non etichettati per imparare a generare immagini da numeri casuali. Queste reti sono cresciute molto negli ultimi anni, specialmente nella generazione di volti sintetici. Il dataset TrueFace presentato in 3.1 raccoglie immagini false generate da queste reti. La creazione di deepfake sarà meglio esplorata in 1.1.2.

In alternativa, il Self Supervised Learning(SSL) è una tecnica di machine learning in cui un modello apprende rappresentazioni significative dei dati senza la necessità di esempi esplicitamente etichettati. L'idea centrale è progettare compiti che il modello può apprendere dai dati non etichettati. Questi compiti sono elaborati per insegnare al modello caratteristiche preziose sui dati in modo indiretto. Non corrispondono direttamente all'obiettivo finale (come il rilevamento dei deepfake in questo caso) ma aiutano il modello a sviluppare abilità utili. La ricostruzione dell'immagine può essere un esempio, dove il modello vede un'immagine corrotta e ha il compito di ricostruire la versione originale. Questo aiuta il modello a comprendere la struttura sottostante e le relazioni dei componenti all'interno dell'immagine. La conoscenza acquisita dal modello pre-addestrato (rappresentazioni delle caratteri-

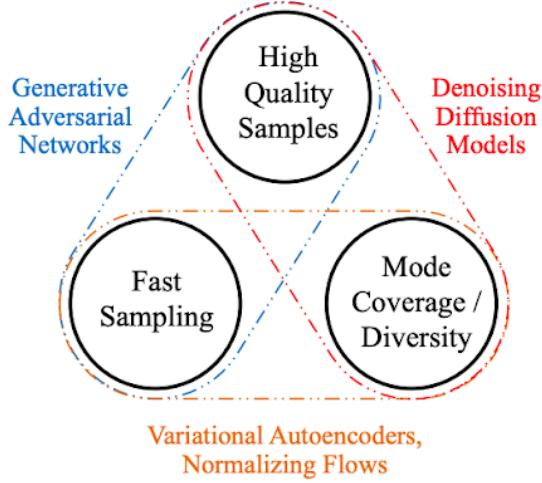


Figura 1.1: Il Dilemma dell’Apprendimento Generativo: attualmente, nessuna tecnica soddisfa tutti e 3 i requisiti. Le GAN mancano di diversità e copertura delle modalità, i Modelli di Diffusione richiedono un elevato tempo di generazione e gli Autoencoder non raggiungono una qualità sufficiente.

stiche) può poi essere applicata a nuovi compiti, come il problema del rilevamento dei deepfake. Il modello può essere ulteriormente allenato sul dataset TrueFace etichettato per adattare le sue conoscenze apprese al compito specifico di classificazione. Sia l’apprendimento Self Supervised che quello Unsupervised coinvolgono l’addestramento di modelli di machine learning senza dati esplicitamente etichettati dall’uomo. Tuttavia, differiscono nel modo in cui utilizzano i dati disponibili perché l’apprendimento Self Supervised può essere visto come una forma di Unsupervised in cui il modello genera le proprie informazioni di supervisione, mentre l’approccio standard mira principalmente a compiti specifici come il clustering, la riduzione della dimensionalità o il rilevamento delle anomalie.

1.1.2 Generazione di Deepfakes

Mentre il termine *deepfake* solitamente si riferisce a contenuti multimediali generati da reti neurali profonde e che rappresentano una persona il cui volto è stato scambiato con le sembianze di qualcun altro, in questo lavoro il termine si riferirà anche a immagini di volti generate da zero. I deepfake si sono evoluti significativamente e sono ora incredibilmente realistici e difficili da rilevare [31]. Per un’ampia adozione nelle applicazioni del mondo reale, i modelli generativi dovrebbero idealmente soddisfare tre requisiti chiave:

- **Campionamento di alta qualità:** Per interagire direttamente con gli utenti, le immagini devono essere dettagliate e di alta qualità per essere indistinguibili da quelle naturali.
- **Copertura delle modalità e diversità dei campioni:** Se i dati di addestramento contengono una vasta quantità di diversità, il modello dovrebbe essere in grado di catturare tale diversità senza sacrificare la qualità della generazione.
- **Campionamento veloce e computazionalmente economico:** Molte applicazioni interattive richiedono un campionamento a bassa latenza, ad esempio, l’editing di immagini in tempo reale.

Questi requisiti costituiscono attualmente il *dilemma dell’apprendimento generativo* [49], poiché non esiste ancora un modello generativo in grado di affrontare tutti e tre i punti contemporaneamente. Tuttavia, alcune ricerche in corso nel campo dei modelli generativi sono focalizzate sull’affrontare questo problema [49]. Mentre gli autoencoder variazionali mostrati nella Figura 1.1 non sono così popolari per la generazione di volti sintetici a causa della loro mancanza di qualità, due tecniche prominenti utilizzate per questo scopo sono le Reti Generative Avversarie (GAN) e i Modelli di Diffusione (DM), che discuteremo in questa sezione.



Figura 1.2: Da [29]. La normalizzazione delle immagini comporta la presenza di artefatti nelle immagini di StyleGAN. Questi non sono sempre evidenti nelle immagini generate, ma sono un problema sistematico per l'architettura stessa del modello.

Generative Adversarial Networks

Una Rete Generativa Avversaria (GAN) è una classe di framework di apprendimento automatico che impiega due reti neurali in un contesto competitivo per facilitare la modellazione generativa. Introdotte da Ian Goodfellow et al. nel 2014 [19], le GAN sono emerse da allora come un approccio prominente all'AI generativa. Il framework GAN comprende due componenti principali: un *generatore* e un *discriminatore*. Il generatore è una rete che sintetizza nuove istanze di dati che assomigliano a un dataset di addestramento. Un generatore mira a produrre output statisticamente indistinguibili dai campioni reali. Dall'altro lato, il discriminatore è utilizzato come classificatore binario per rilevare se un punto dati è l'output del generatore o un'istanza del dataset. Agisce come un 'critico', fornendo feedback al generatore riguardo all'autenticità dei suoi output. Il processo di addestramento comporta un gioco iterativo in cui il generatore e il discriminatore si impegnano in un costante ciclo di feedback. Il generatore si sforza di produrre dati sempre più realistici per ingannare il discriminatore, mentre il discriminatore affina continuamente la sua capacità di distinguere tra dati reali e sintetici. Attraverso questa interazione competitiva, entrambe le reti migliorano progressivamente le loro prestazioni, convergendo infine verso uno stato in cui il generatore produce output altamente convincenti che il discriminatore fatica a identificare come falsi. A questo punto, il generatore viene utilizzato per produrre nuovi campioni, mentre il discriminatore viene scartato. Tra i molti tipi di GAN disponibili, StyleGAN [27] è un'architettura sviluppata dai ricercatori di Nvidia. Introdotta per la prima volta nel 2018, ha guadagnato una notevole attenzione per la sua capacità di generare immagini di alta qualità e realistiche, in particolare di volti. StyleGAN è la prima architettura GAN capace di generare immagini con una risoluzione di 1024 pixel. Dal suo rilascio iniziale nel 2019, StyleGAN ha subito diversi miglioramenti e iterazioni. StyleGAN2 [29] ha affrontato alcuni degli artefatti presenti nel modello originale e mostrati nella Figura 1.2, mentre StyleGAN3 [26] ha ulteriormente migliorato la capacità del modello di generare immagini coerenti e realistiche.

La particolarità della famiglia di reti si basa sulla loro capacità di controllare lo stile delle immagini generate. Il concetto di stile deriva da una tecnica di deep learning chiamata *style transfer*, che combina il contenuto di un'immagine con lo stile artistico di un'altra, creando un risultato che fonde entrambi gli elementi. Nello style transfer, questo si ottiene utilizzando due diverse CNN, una per estrarre la rappresentazione del contenuto e un'altra per la rappresentazione dello stile. I risultati di queste reti possono poi essere riassemblati, scambiando lo stile di un'immagine con il contenuto di un'altra. Sebbene questa non sia la tecnologia utilizzata nelle StyleGAN, il concetto di stile è utilizzato in modo simile. Queste reti sfruttano un *vettore di stile*, che viene utilizzato come modo per ottenere il controllo sugli attributi di stile come il colore dei capelli, l'aggiunta di rughe o la modifica delle espressioni facciali. Il vettore di stile è il risultato di una proiezione iniziale del classico rumore casuale z in uno spazio latente intermedio \mathcal{W} con una rete di mappatura f in modo che il vettore di stile $w = f(z)$, $z \in \mathcal{Z}$, $w \in \mathcal{W}$. Il vettore di stile può essere passato a diversi strati della rete GAN tradizionale per implementare una certa caratteristica di stile. In questa fase, una versione scalata e traslata del vettore di stile viene passata utilizzando un'operazione chiamata normalizzazione adattiva dell'istanza (AdaIN), ottenendo un controllo dettagliato sulle caratteristiche. Inoltre, le StyleGAN introducono una variazione stocastica aggiungendo rumore casuale al vettore di stile in ogni strato. Questa casualità contribuisce alla diversità delle immagini generate e permette la creazione di

variazioni uniche dello stesso stile. Un esempio dell’architettura StyleGAN confrontata con una GAN standard è presentato nella Figura 1.3.

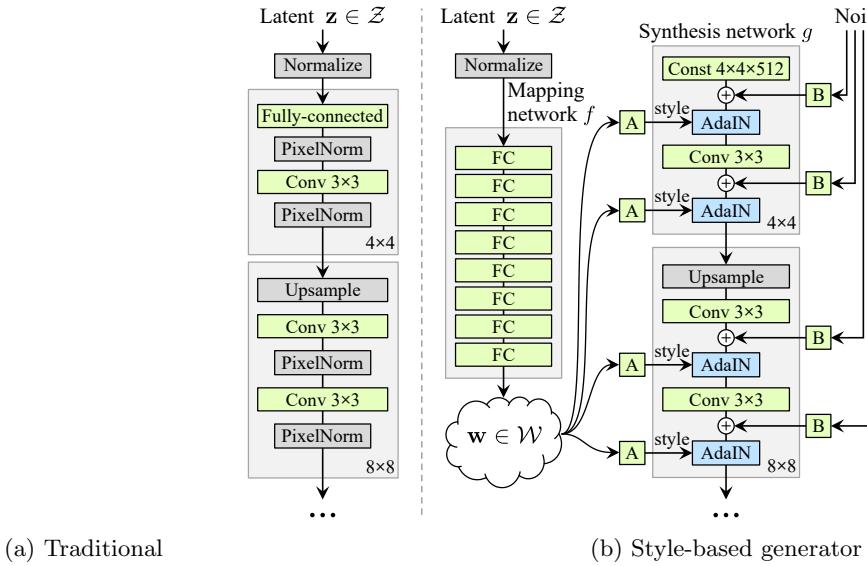


Figura 1.3: Da [28]. Mentre un generatore tradizionale [25] alimenta il codice latente solo attraverso il livello di input, in [28] prima mappano l’input in uno spazio latente intermedio \mathcal{W} , che poi controlla il generatore attraverso la normalizzazione adattiva dell’istanza (AdaIN) in ogni strato di convoluzione. Il rumore gaussiano viene aggiunto dopo ogni convoluzione, prima di valutare la non linearità. Qui “A” rappresenta una trasformazione affine appresa, e “B” applica fattori di scala appresi per canale all’input del rumore. La rete di mappatura f consiste di 8 strati e la rete di sintesi g consiste di 18 strati - due per ogni risoluzione ($4^2 - 1024^2$). L’output dell’ultimo strato viene convertito in RGB usando una convoluzione 1×1 separata, simile a Karras et al. [25].

Diffusion Models

I Modelli di Diffusione (Diffusion Models - DM) sono un altro tipo di modello generativo che recentemente è diventato molto popolare per la creazione di dati di alta qualità come immagini, video, audio e altro. Grazie alla loro versatilità, i DM sono già stati applicati a una varietà di compiti di generazione, come la sintesi di immagini, audio, forme 3D e grafi. Il funzionamento di un modello di diffusione si basa su due fasi principali: il processo di diffusione in avanti e il processo di diffusione inverso. Il processo di diffusione in avanti mappa i dati in rumore perturbando gradualmente i dati di input. Questo viene formalmente ottenuto da un semplice processo stocastico che parte da un campione di dati e genera iterativamente campioni più rumorosi utilizzando un semplice kernel di diffusione gaussiano. In altre parole, ad ogni passo di questo processo, il rumore gaussiano viene incrementalmente aggiunto ai dati, fino a quando l’immagine originale viene completamente trasformata in puro rumore. Il secondo processo è un processo inverso parametrizzato che annulla la diffusione in avanti ed esegue una riduzione del rumore iterativa. Questo processo rappresenta la sintesi dei dati ed è addestrato a generare dati convertendo il rumore casuale in dati realistici. È anche formalmente definito come un processo stocastico, che opera iterativamente sulle immagini di input utilizzando reti neurali profonde addestrabili. I processi in avanti e inverso spesso utilizzano migliaia di passaggi per l’iniezione graduale di rumore e durante la generazione per la riduzione di rumore.

Sebbene i modelli di diffusione non siano famosi in particolare per la generazione di volti sintetici, sono generalmente in grado di superare le prestazioni delle GAN più vecchie. Tuttavia, questi modelli sono costosi in termini di tempo per generare un nuovo campione, a causa dei ripetitivi passaggi di denoising. Uno dei modelli di diffusione più noti è DALL-E 2, sviluppato da OpenAI, che può generare immagini da descrizioni testuali. Questo modello ha dimostrato una straordinaria capacità di comprendere il linguaggio naturale e tradurre le parole in immagini realistiche e creative.

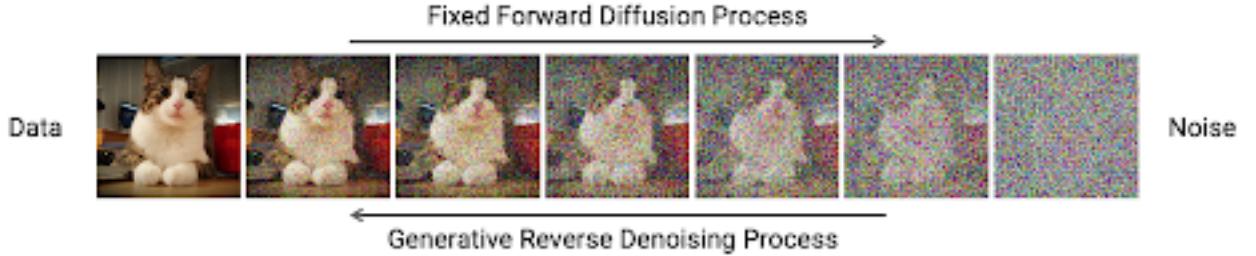


Figura 1.4: Processi del modello di diffusione che si muovono da e verso i dati e il rumore

1.2 Rilevazione di Deepfakes

Il rilevamento di immagini sintetiche, specialmente quelle generate da GAN e DM, si basa fortemente sull’identificazione di tracce sottili o artefatti incorporati nell’immagine durante il processo di generazione. Queste ‘tracce forensi’, siano esse anomalie visibili o pattern latenti nel dominio delle frequenze, fungono da impronte digitali che distinguono le immagini sintetiche da quelle catturate da dispositivi del mondo reale. Questa sezione esamina i metodi all’avanguardia che sfruttano queste tracce, esplorando sia le analisi nel dominio spaziale che in quello delle frequenze, e approfondendo le sfide di generalizzazione e robustezza di fronte all’evoluzione delle architetture di generazione e alle manipolazioni delle immagini. Specialmente quando i modelli generativi erano meno sofisticati, alcuni lavori si sono concentrati principalmente sull’incapacità di tali generatori di riprodurre perfettamente le caratteristiche semantiche di alto livello delle immagini naturali [21]. Possono produrre artefatti visibili, come anomalie cromatiche (Figura 1.2) o mancanza di simmetrie naturali. Ad esempio, i volti generati potrebbero avere occhi sinistro e destro di colore diverso, riflessi oculari poco convincenti o forme irregolari delle pupille [21]. Per i DM [12] in particolare, è stato notato che la mancanza di una modellazione 3D esplicita di oggetti e superfici causa asimmetrie nelle ombre e nelle immagini riflesse. Inoltre, si può osservare, in una certa misura, un’incoerenza semantica globale nell’illuminazione. Tuttavia, nuovi metodi basati su DM o GAN possono superare queste limitazioni [48] e generare immagini che soddisfano tutti i vincoli semantici necessari, sia per quanto riguarda l’illuminazione, la prospettiva o qualsiasi altro aspetto.

Per questo motivo, la maggior parte dei rilevatori all’avanguardia di immagini false si basa su tracce invisibili all’occhio umano [20]. Immagini visivamente perfette, senza evidenti incoerenze semantiche, possono essere distinte da immagini reali basandosi sulle tracce intrinseche al processo di generazione. Qualsiasi metodo utilizzato per creare dati visivi sintetici incorpora alcune tracce peculiari [8] nelle immagini di output, che sono correlate alle azioni intraprese nel processo di generazione. Inoltre, ogni architettura generativa è caratterizzata dalle sue tracce peculiari, consentendo così anche l’attribuzione della fonte. La presenza e la distintività di tali tracce sono state dimostrate estraendo una sorta di impronte digitali artificiali nel dominio spaziale, ma anche attraverso analisi nel dominio delle frequenze che mostrano come l’operazione di upsampling eseguita nella maggior parte delle architetture GAN dia origine a chiari picchi spettrali. Il complesso processo di elaborazione necessario per generare immagini sintetiche porta inevitabilmente all’introduzione di artefatti digitali che caratterizzano l’architettura specifica e differiscono significativamente da quelli tipici dei dispositivi di acquisizione tradizionali. Nel tentativo di sfruttare gli artefatti nel dominio spaziale, sono state testate diverse tecniche. Ad esempio, [37] sfrutta il fatto che le GAN producono solo una gamma limitata di valori di intensità e non generano regioni saturate e/o sottoesposte. Allo stesso modo, [32] sfrutta l’incapacità delle GAN di preservare accuratamente la correlazione naturale tra le bande di colore. Di conseguenza, per estrarre caratteristiche discriminative per il rilevamento, le componenti di crominanza vengono filtrate passa-alto e riassunte dalle loro matrici di co-occorrenza. Una matrice di co-occorrenza è uno strumento statistico utilizzato per analizzare la texture di un’immagine. Quantifica la frequenza con cui coppie di pixel con valori specifici e in una relazione spaziale specifica si verificano in un’immagine. Vale la pena notare che le co-occorrenze di versioni filtrate passa-alto dell’immagine sono strumenti popolari

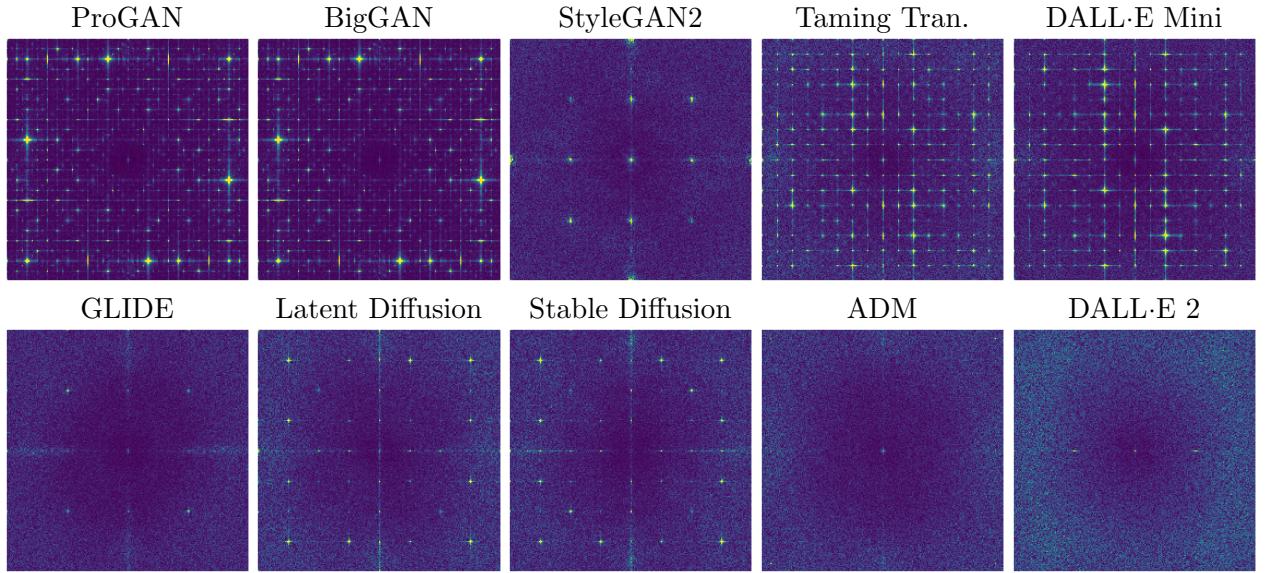


Figura 1.5: Da [8]. Trasformata di Fourier (ampiezza) dell’impronta artificiale stimata da 1000 residui di immagine. In alto: da sinistra a destra ProGAN [25], BigGan [5], StyleGAN2 [30], Taming Transformers [15], DALL·E Mini [11]. Sotto: GLIDE [39], Latent Diffusion [44], Stable Diffusion [45], ADM [12], DALL·E 2 [43].

nella forense delle immagini poiché gli artefatti invisibili sono spesso presenti nelle componenti del segnale ad alta frequenza [47]. Le matrici di co-occorrenza estratte dai canali RGB sono utilizzate anche in [16] come input di una CNN.

In alternativa, il dominio delle frequenze fornisce informazioni utili per il rilevamento, poiché le immagini GAN mostrano chiare tracce della loro origine sintetica nel dominio di Fourier. Il rilevatore proposto in [51] sfrutta la presenza di picchi spettrali causati dalle operazioni di upsampling eseguite di routine nella maggior parte delle architetture GAN. L’analisi nel dominio delle frequenze viene effettuata anche in [17] per studiare la presenza di artefatti in diverse architetture di rete (Figura 1.5), dataset e risoluzioni. Ancora una volta, questi artefatti vengono utilizzati per distinguere le immagini generate da quelle reali. In particolare, un classificatore basato su CNN viene addestrato con spettri di Fourier presi sia da immagini reali che dalle loro versioni sintetiche ottenute attraverso un autoencoder avversario. Allo stesso modo, in [14] si dimostra che le immagini GAN non imitano fedelmente le distribuzioni spettrali delle immagini naturali. Viene proposto un semplice rilevatore che prende la distribuzione dell’energia spettrale come caratteristica di input. Gli autori propongono anche una perdita spettrale da utilizzare durante l’addestramento GAN per limitare l’apparizione di artefatti spettrali.

Tuttavia, gli approcci completamente supervisionati descritti sopra sono tutti molto efficaci quando le immagini GAN in esame provengono da un modello che è anche presente nell’addestramento, ma falliscono nel generalizzare a dati generati da nuovi modelli mai visti. Pertanto, alcuni metodi sono stati proposti recentemente per affrontare questo problema. In [9, 13] vengono proposte strategie di apprendimento con pochi esempi, con un’architettura basata su autoencoder, per adattarsi a nuove manipolazioni con solo pochi esempi. In [35], invece, viene utilizzato un approccio basato sull’apprendimento incrementale. Nonostante la migliore generalizzazione, questi metodi hanno ancora bisogno di alcuni esempi della nuova architettura GAN, il che non è sempre realistico. Una soluzione diversa è proposta in [50]. L’idea è di effettuare un’augmentation mediante sfocatura gaussiana per costringere il discriminatore ad apprendere caratteristiche più generali. Un approccio simile è seguito in [48] dove una ResNet50 pre-addestrata viene ulteriormente allenata con una forte augmentation basata su compressione e sfocatura. Gli esperimenti mostrano che, anche addestrando su una singola architettura GAN, le caratteristiche apprese si generalizzano bene ad architetture, dataset e metodi di addestramento mai visti. Una prospettiva diversa è adottata in [6] dove viene proposto un classificatore completamente convoluzionale basato su patch. Gli autori mostrano che concentrandosi su patch locali piuttosto che su strutture globali, possono ottenere prestazioni migliori.

Basandosi su questi concetti, sono stati sviluppati diversi promettenti rilevatori di immagini GAN basati su CNN. Tuttavia, il problema è lungi dall'essere risolto. Da un lato, nuove e più sofisticate architetture di generazione vengono proposte ogni giorno, alcune di esse, come StyleGAN3 [26], mirano esplicitamente a minimizzare la presenza di queste tracce indesiderate. D'altro lato, i rilevatori subiscono un significativo calo delle prestazioni quando la qualità dell'immagine è compromessa [3], come accade sempre sui social network, che applicano di routine alcune operazioni di ridimensionamento e compressione. Questo perché le tracce forensi sono delicate e possono essere facilmente rimosse anche da passaggi di elaborazione non malevoli come compressione, ridimensionamento o taglio.

La riproducibilità di queste tecniche su piattaforme con risorse limitate rimane una questione aperta e un focus centrale di questo lavoro. Per affrontare questo, la sezione successiva si addentra nell'architettura e nella funzionalità di due importanti famiglie di reti progettate per tali piattaforme: PhiNets [40] e MicroNet [33].

1.3 Architetture per dispositivi a risorse limitate

L'Internet delle Cose (Internet of Things - IoT) è un mondo di dispositivi intelligenti interconnessi, sia mobili che fissi. Per gestire la vasta quantità di dati che generano, dobbiamo spostare l'intelligenza lontano dai cloud centralizzati e più vicino ai dispositivi stessi, al "bordo" della rete. Tuttavia, questa elaborazione di bordo presenta una sfida. Limitazioni di risorse come bassa memoria, capacità di comunicazione ristrette e budget di batteria limitato rendono l'esecuzione di algoritmi di AI tradizionali su questi dispositivi periferici, come i nodi di rete di sensori, piuttosto difficile. Ciò richiede metodi innovativi per portare l'AI al bordo. In questa sezione, analizzeremo le principali limitazioni in termini di architetture con risorse limitate, analizzando le due architetture che saranno utilizzate in questo lavoro: una backbone scalabile adatta ai microcontrollori (MicroController Unit - MCU) chiamata PhiNet[40] e una famosa architettura per TinyML chiamata Micronet [33].

1.3.1 Limitazioni su Micro Controllori

Quando si lavora su microcontrollori, le risorse sono particolarmente preziose. Ad esempio, un MCU STM32H743 ha $2MB$ di Flash interna e $1MB$ di RAM, e la tensione di ingresso è solitamente intorno a pochi Volt. Questi requisiti sono importanti e dipendono dal microcontrollore. Quando si sviluppa un'architettura edge che deve essere implementata su un MCU, questi sono i principali fattori che verranno utilizzati.

- **MAC** sta per Multiply Accumulate ed è la quantità di operazioni richieste dal modello per eseguire il passo di inferenza. È una delle operazioni predefinite che un'unità di elaborazione è in grado di fare ed è presa come riferimento perché costituisce la maggior parte del carico computazionale nel caso di reti neurali convoluzionali, sono usate per moltiplicare i pesi di un kernel con i valori di attivazione e accumulare i risultati.
- **Utilizzo della RAM** è un vincolo importante quando si implementano reti neurali su dispositivi edge. Si collega direttamente alla dimensione dei tensori più grandi richiesti durante il passo di inferenza della rete. Anche il motore di inferenza scelto gioca un ruolo. Tuttavia, i miglioramenti relativi nell'utilizzo della RAM per un motore fisso rimangono significativi, rendendo i confronti tra operatori validi e generalizzabili tra diversi motori.
- **Conteggio dei parametri** impone un vincolo rigido su quali piattaforme possono ospitare una specifica rete neurale. Questo perché il conteggio dei parametri è direttamente correlato alla quantità di memoria FLASH utilizzata.
- **Intensità aritmetica** misura il numero medio di operazioni aritmetiche eseguite per byte caricato in un registro della CPU. Per una quantità fissa di calcolo, una maggiore intensità aritmetica si traduce in costi di accesso alla memoria inferiori. Ciò rende l'operazione meno vincolata dalla memoria e, in definitiva, più efficiente.

1.3.2 PhiNets

L'architettura di Phinets consiste in una classe di reti scalabili ottimizzate per l'elaborazione di immagini basata su deep learning su piattaforme con risorse limitate [41]. In particolare per i vari compiti

di classificazione delle immagini, rilevamento e tracciamento di oggetti in tempo reale, le reti convoluzionali di questa famiglia si distinguono per la struttura dell'ultimo strato (*classificatore* o *testa di rilevamento*), mantenendo una *backbone* comune, che è l'insieme degli strati convoluzionali utilizzati per l'estrazione delle caratteristiche. Per rendere questi modelli scalabili, è essenziale ottimizzare la backbone della rete poiché è responsabile della maggior parte del consumo di risorse durante l'inferenza del modello. Le tecniche utilizzate a questo scopo coinvolgono principalmente lo sfruttamento di blocchi convoluzionali efficienti e l'uso di un'architettura che può essere modificata attraverso appropriati fattori di scala.

Le architetture PhiNets sono particolarmente ottimizzate in termini di requisiti computazionali grazie all'uso di un tipo specifico di convoluzione in ogni strato. Infatti, i *blocchi residuali invertiti* si sono dimostrati particolarmente efficaci a questo scopo fin dalla loro introduzione in MobileNetV2 [46].

Questo tipo di blocco è una versione diversa di un famoso blocco convoluzionale chiamato *blocco residuale*. Questi blocchi residuali prendono il nome dall'operazione di *connessione saltata* che collega l'input e l'output dell'operazione convoluzionale, che propaga residui della rappresentazione di input. Questo viene fatto per consentire il riutilizzo delle caratteristiche di input nei livelli successivi e affrontare il problema dei *gradienti che svaniscono*. È stato notato che i gradienti diventano più piccoli quando la rete è più profonda, rendendo difficile per il modello apprendere efficacemente. La connessione saltata permette ai gradienti di fluire direttamente attraverso il blocco, rendendo più facile l'addestramento di reti più profonde. L'operazione di base per il blocco residuale implica la riduzione del numero di canali nella mappa delle caratteristiche in entrata attraverso un'operazione convoluzionale economica, seguita dall'espansione del numero di canali (riportato al valore iniziale), e l'applicazione della connessione saltata, che lascia residui della rappresentazione di input nella mappa delle caratteristiche di output.

Come alternativa, sono stati introdotti i *blocchi residuali invertiti*. La procedura operativa è molto simile a quella dei blocchi residuali tradizionali, con la differenza che il numero di canali nella rappresentazione di input viene inizialmente espanso per estrarre informazioni più dettagliate, e poi compresso alla fine del blocco per ottenere una rappresentazione di output con la stessa dimensionalità dell'input. I blocchi residuali invertiti sono particolarmente adatti alle architetture edge grazie all'uso di convoluzioni speciali chiamate convoluzioni point-wise e depth-wise. Dato un tensore con C canali che deve essere mappato su un tensore con C' canali, una convoluzione standard utilizza un kernel di dimensione k^2C e l'intera operazione di mappatura richiede k^2CC' parametri. Questo è il tipo di convoluzione che viene eseguita, ad esempio, nei blocchi residuali di una ResNet50. Nelle architetture edge, la convoluzione standard viene "fattorizzata" in una convoluzione point-wise e una depth-wise. La convoluzione point-wise si occupa della comunicazione inter-canale su un singolo pixel alla volta e permette al modello di espandere o comprimere il numero di canali. D'altra parte, la convoluzione depth-wise si concentra solo su un singolo canale del tensore di input alla volta, ma opera nel dominio spaziale.

Con questo metodo, il kernel ha una dimensione di k^2 (k^2C operazioni) per la convoluzione depth-wise e C (CC' operazioni) per la convoluzione point-wise. La riduzione della complessità è il motivo per cui i blocchi residuali invertiti sono molto migliori dei blocchi residuali in termini di numero di operazioni e parametri. Lo svantaggio di questo approccio risiede nel fatto che una convoluzione 2D può operare sulla risoluzione spaziale su diversi canali simultaneamente, il che di solito porta a prestazioni più elevate.

A questa struttura di base del blocco residuale invertito, l'architettura PhiNets aggiunge un'operazione chiamata *squeeze and excitation* prima della compressione finale del numero di canali. Questa funzione aiuta a enfatizzare le informazioni presenti nei canali del tensore di input prima di eseguire la compressione.

Le architetture PhiNets adottano l'uso di determinati fattori di scala per adattare la struttura e il consumo di risorse della rete alla specifica piattaforma utilizzata. In particolare, PhiNets utilizza 4 diversi iperparametri per modificare la struttura del modello:

- α (*fattore di larghezza*): Regola la larghezza della rete modificando il numero di canali di output da ciascuno strato.

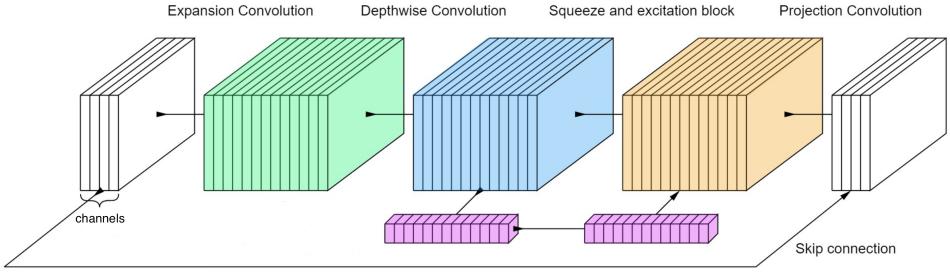


Figura 1.6: Una panoramica della struttura del blocco convoluzionale di PhiNets. Prima, il numero di canali viene aumentato con una convoluzione point-wise, seguita da una convoluzione depth-wise e un blocco SE. Infine, una seconda convoluzione point-wise si collega al blocco bottleneck a bassa dimensionalità.

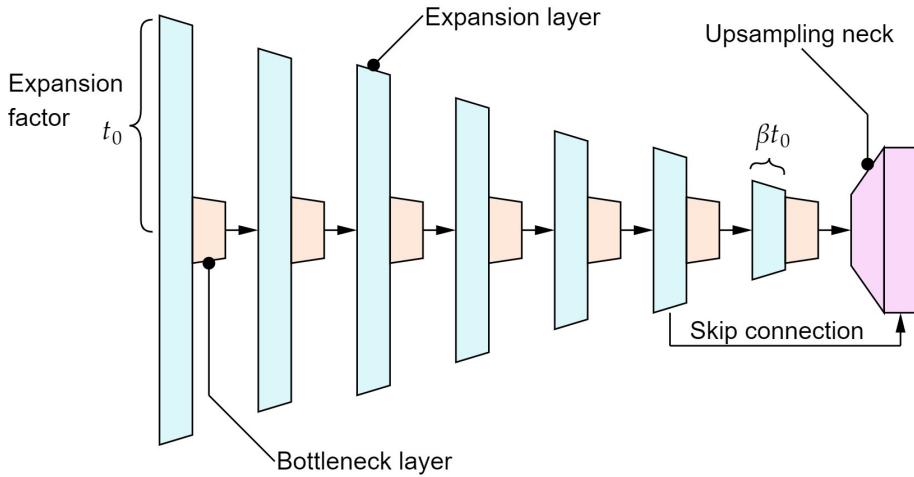


Figura 1.7: Struttura del corpo di PhiNets. Ogni iperparametro α , β , t_0 e N influenza la struttura generale.

- β (*fattore di forma*): Modifica la risoluzione spaziale delle immagini in ciascuno strato convoluzionale.
- N : è il numero totale di strati, che determina la profondità della rete.
- t_0 (*fattore di espansione di base*): Controlla il numero di canali nelle rappresentazioni espansse durante l'esecuzione di ciascun blocco residuale invertito. In particolare, il *fattore di espansione* t varia linearmente con la profondità della rete secondo la formula $t = t_0 \left(\frac{(\beta-1)n+N}{N} \right)$, dove n è l'indice del blocco convoluzionale a cui ci riferiamo.

Sfruttando questi parametri, è possibile avere il pieno controllo della struttura del modello e dell'utilizzo delle risorse (Figura 1.7). In particolare, il numero di MAC dipende dalla risoluzione dell'immagine $w \times h$, dalla profondità della rete (N) e dalla sua larghezza, che scala quadraticamente con α . Il numero di parametri è determinato dalla dimensione dei kernel nei blocchi convoluzionali, che aumenta linearmente con N e β .

1.3.3 MicroNet

MicroNet è un'altra famiglia di reti che mira ad affrontare il problema del sostanziale degrado delle prestazioni quando è richiesto un costo computazionale minuscolo [33]. Questo viene ottenuto con due caratteristiche tecniche. In primo luogo, la convoluzione micro-fattorizzata è un metodo che fattorizza una matrice di convoluzione in matrici di rango inferiore, permettendo al sistema di eseguire meno operazioni. Questo avviene sia per le convoluzioni point-wise che per quelle depth-wise, combinando poi le due tecniche. In entrambi i casi, la formula generale descrive il kernel iniziale \mathbf{W} come un prodotto di tre matrici più piccole:

$$\mathbf{W} = \mathbf{P}\Phi\mathbf{Q}^T \quad (1.1)$$

Per esempio, il kernel di convoluzione depth-wise $k \times k$ può essere fattorizzato in un kernel $k \times 1$ e un kernel $1 \times k$. Questo segue l'Eq. 1.1, con la matrice del kernel $k \times k$ per canale \mathbf{W} , il vettore $k \times 1$ \mathbf{P} , il vettore $1 \times k$ \mathbf{Q}^T e Φ uno scalare di valore 1. Questa approssimazione di rango inferiore riduce la complessità computazionale da $\mathcal{O}(k^2C)$ a $\mathcal{O}(kC)$.

Inoltre, viene utilizzata una nuova funzione di attivazione chiamata *Dynamic Shift Max*. Il nome *dynamic* si riferisce al fatto che la funzione utilizza pesi che dipendono dall'input, contrariamente alle reti statiche. Questa funzione è progettata per migliorare la non linearità attraverso i seguenti passaggi:

- **Spostamento circolare dei canali:** DSM prende una mappa di feature di input e crea molteplici versioni spostate di essa spostando circolarmente gruppi di canali.
- **Fusione dinamica:** Queste mappe di feature spostate vengono poi combinate con la mappa di feature di input originale in un modo che viene appreso dalla rete durante l'addestramento e dipende dall'input. Consiste in una sequenza di average pooling, due strati completamente connessi e uno strato sigmoid, come nel blocco Squeeze-and-Excitation.
- **Maxing Out:** Infine, DSM applica l'operazione di massimo su tutte le mappe di feature fuse, selezionando le caratteristiche più salienti da ciascuna.

Sfruttando queste due tecniche è possibile creare i *Micro-Blocks*, che sono una combinazione di convoluzioni depthwise, convoluzioni pointwise e la funzione di attivazione Dynamic Shift Max. Vengono quindi creati i Micro-Block A, B e C, che costituiscono il blocco di costruzione per MicroNet. Maggiori dettagli possono essere trovati nel documento ufficiale [33].

Dato il potenziale di PhiNet e MicroNet, questo lavoro esplora la loro capacità di ottenere prestazioni paragonabili a ResNet50 [22] nel compito di rilevamento dei deepfake. Affrontare questo problema permetterebbe tempi di inferenza più rapidi e un minor consumo energetico, consentendo controlli di validità delle immagini in tempo reale. Inoltre, una tecnica di apprendimento auto-supervisionato mira a migliorare le prestazioni di questi modelli su immagini compresse, dove i modelli standard spesso faticano.

2 Metodi

In questo studio, abbiamo esplorato due approcci distinti per il rilevamento dei deepfake e l'analisi delle immagini. Il primo approccio, dettagliato nella sezione 2.1, adotta una metodologia tradizionale di Apprendimento Supervisionato. Questo coinvolge l'addestramento di modelli su un dataset di immagini Pre-Social, seguito da una cruciale fase di fine-tuning su dati Post-Social media. Questa metodologia, sebbene ben consolidata, sarà valutata criticamente per valutarne i punti di forza, le debolezze e l'idoneità per l'implementazione in scenari reali non controllati. Sequenzialmente, la sezione 2.2 introduce un approccio più innovativo. Qui, esploriamo il potenziale dell'Apprendimento Auto-Supervisionato, utilizzando specificamente il framework SimCLR proposto da Chen et al. [7]. Questo approccio mira a aggirare la necessità di fine-tuning Post-Social sfruttando la struttura intrinseca dei dati Pre-Social durante il processo di addestramento. Indagheremo l'efficacia di questo approccio auto-supervisionato in confronto al metodo supervisionato tradizionale. Per garantire una valutazione completa, implementeremo entrambi gli approcci su una varietà di architetture di modelli. Questo includerà le due architetture all'avanguardia presentate nella Sezione 1.3, così come un modello ResNet50 standard, la cui architettura è rappresentata nella Figura 2.1. Questa analisi comparativa fornirà informazioni preziose sulle prestazioni e la generalizzabilità di ciascun approccio su diverse complessità e architetture di modelli. Esplorando queste diverse metodologie e architetture di modelli, miriamo a contribuire alla ricerca in corso nel rilevamento dei deepfake. Crediamo che i risultati di questo studio saranno di interesse sia per i ricercatori che per i professionisti che lavorano in questo campo.

2.1 Apprendimento Supervisionato Pre-Social e Fine Tuning Post-Social

Questa sezione esplora un approccio in due fasi che utilizza l'apprendimento supervisionato su contenuti pre-social e il fine-tuning per l'analisi di immagini post-social. Il pipeline coinvolge l'utilizzo di modelli pre-addestrati e il loro adattamento al compito specifico di distinguere le immagini reali da quelle false sulle piattaforme di social media. La prima fase inizia con l'utilizzo di architetture di deep learning, come ResNet, PhiNet o XiNet, pre-addestrate su un dataset di immagini su larga scala come ImageNet. ImageNet contiene milioni di immagini etichettate da esseri umani, permettendo al modello di apprendere rappresentazioni generiche di immagini, che possono essere riutilizzate per compiti di elaborazione delle immagini [2, 23]. Durante il pre-addestramento, il modello sviluppa una backbone che codifica un'immagine in un *vettore di feature*, seguito da una testa che classifica l'immagine in una delle migliaia di categorie all'interno del dataset. La fase di pre-addestramento si concentra sulla co-

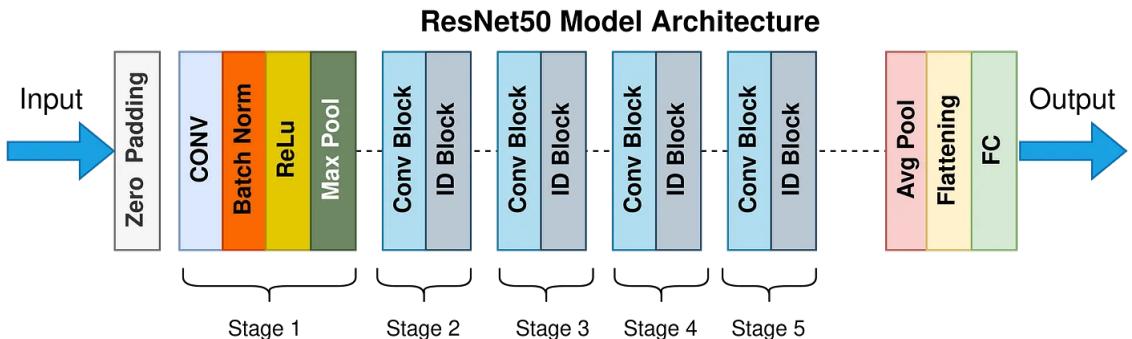


Figura 2.1: L'architettura di una ResNet50 standard. Il fully connected (FC) viene rimosso e viene utilizzata una testa di proiezione o un nuovo strato, a seconda del metodo che stiamo testando.

struzione di una solida base per la comprensione del contenuto visivo. Dopo questa fase, la testa viene scartata e i parametri della backbone vengono congelati. Il congelamento impedisce che la conoscenza della backbone venga sovrascritta durante le successive fasi di addestramento, che è un problema noto come dimenticanza catastrofica [18, 36]. Nel machine learning tradizionale, un modello viene solitamente addestrato da zero per ogni nuovo compito. A volte, il *Transfer Learning* [52] può essere utilizzato per ridurre il tempo di addestramento e ottenere prestazioni più elevate. Questa tecnica sfrutta il fatto che molti compiti condividono caratteristiche o pattern comuni, e consiste nello sfruttare un modello che è stato addestrato su un compito per riutilizzarlo per un secondo compito correlato. In questo approccio, i modelli sono stati pre-addestrati su ImageNet per la classificazione delle immagini e le loro abilità di codifica vengono trasferite per il compito di rilevamento dei deepfake. La seconda fase coinvolge l'uso della backbone congelata come base per una nuova testa specificamente progettata per il compito di classificare immagini reali e false da TrueFace. Poiché la backbone rimane congelata, questa fase si concentra esclusivamente sull'addestramento della nuova testa per la classificazione binaria basata sui vettori di feature estratti dalla backbone. Qui, l'addestramento viene eseguito su immagini pre-social, che non sono state condivise su alcuna piattaforma di social media. Una volta che la testa è stata addestrata, il modello viene valutato su immagini condivise su piattaforme come Twitter, Facebook e Telegram. Questa fase di valutazione valuta come i processi di compressione delle immagini impiegati da queste piattaforme potrebbero influenzare le prestazioni del modello. Questo passaggio è cruciale per comprendere le potenziali limitazioni del modello quando applicato a scenari di utilizzo su social media nel mondo reale. La potenziale mancanza di generalizzazione su tutte le piattaforme di social media è riconosciuta come una limitazione. Per affrontare questo problema, un passaggio finale coinvolge la creazione di copie della testa addestrata e il loro fine-tuning su dataset di immagini post-social specifici per ogni piattaforma (Twitter, Facebook, Telegram). Questo processo di fine-tuning mira a colmare eventuali gap di prestazioni identificati durante la fase di valutazione e a migliorare la capacità del modello di gestire le sfumature delle immagini da ciascuna piattaforma di social media. In questo modo, creiamo un limite superiore per le prestazioni nei tre diversi contesti, che può essere utilizzato come punto di confronto per diverse tecniche.

Funzione di perdita Poiché il rilevamento dei deepfake è un problema di classificazione binaria (reale o falso), la funzione di perdita appropriata è l' Entropia Incrociata Binaria (Binary Cross Entropy - BCE loss) . Essa misura la differenza tra le probabilità previste dal modello per la classe reale e le etichette effettive (0 per falso, 1 per reale) nei dati di addestramento. Dato un punto dati, sia y l'etichetta binaria vera, e sia $\hat{y} = f(x)$ la probabilità prevista per la classe vera y prodotta dal modello (tra 0 e 1) per un dato input x . La BCE Loss per questo punto dati è:

$$\mathcal{L}(\hat{y}, y) = -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y}) \quad (2.1)$$

Il termine $\log(\hat{y})$ rappresenta il contributo alla perdita per la classe vera. Una probabilità prevista più alta porta a una perdita inferiore per questo termine nel caso di un'etichetta positiva. Simmetricamente, il termine $\log(1 - \hat{y})$ rappresenta il contributo alla perdita per la classe falsa. Sebbene la funzione di perdita BCE si aspetti che la probabilità prevista dal modello cada nell'intervallo $[0, 1]$, l'output del modello è inizialmente sotto forma di un *logit*, un numero reale sulla linea dei numeri infinita. Per colmare questo divario, viene introdotta la funzione sigmoid. Questa funzione agisce come un traduttore, trasformando i valori logit in probabilità adatte per la perdita BCE. La funzione sigmoid è definita come segue:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (2.2)$$

Ad esempio, un valore logit di 0 viene mappato a 0,5 dalla funzione sigmoid. Questo rappresenta uno stato di completa incertezza per il modello, dove non può dire con sicurezza se l'immagine è reale o falsa. Man mano che il valore logit aumenta verso l'infinito positivo, la funzione sigmoid si avvicina a 1. Questo significa che il modello diventa sempre più sicuro che l'immagine sia reale.

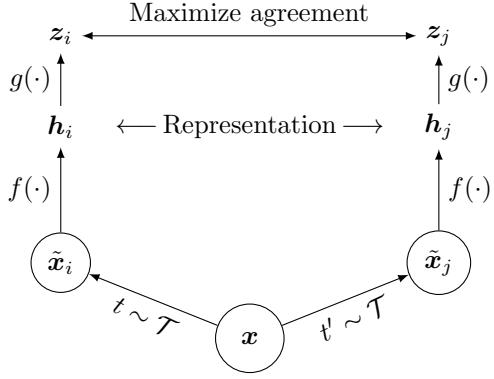


Figura 2.2: Da [7]. Un Semplice Framework per l’Apprendimento Contrastivo di Rappresentazioni Visive. Due diversi moduli di augmentation vengono estratti dalla stessa famiglia di augmentation ($t \sim \mathcal{T}$ and $t' \sim \mathcal{T}$) e applicati su ognuna delle due copie di una stessa immagine di partenza. Un encoder di base $f(\cdot)$ e una testa di proiezione $g(\cdot)$ sono allenate per massimizzare la similità usando una funzione di perdita contrastiva. Al termine dell’allenamento, la testa di proiezione $g(\cdot)$ viene scartata e l’encoder $f(\cdot)$ viene utilizzato per il task successivo.

2.2 Apprendimento Auto-Supervisionato con Fine Tuning Supervisionato

Il secondo approccio affronta il rilevamento dei deepfake sui social media dando priorità alla robustezza contro la compressione delle immagini fin dall’inizio, potenzialmente eliminando la necessità di un fine-tuning specifico per i social media in seguito. Dopo aver sfruttato il transfer learning in modo simile all’ultimo approccio, la seconda fase introduce l’addestramento auto-supervisionato utilizzando la backbone pre-addestrata. Lo fa utilizzando un *pretext task* chiamato *apprendimento contrastivo* (CL), che consiste nell’addestrare un modello ad avvicinare le rappresentazioni di punti dati simili (coppie positive) mentre allontana le rappresentazioni di punti dati dissimili (coppie negative). Mentre questo è un approccio generale all’apprendimento auto-supervisionato e è adattabile a audio, immagini, testo e vari tipi di dati, questa fase impiega un’implementazione di CL chiamata SimCLR, un framework di apprendimento contrastivo (ulteriormente esplorato nella sottosezione 2.2.1) per addestrare ulteriormente la backbone. Questo addestramento si concentra su un compito che è agnostico al rilevamento dei deepfake ma incoraggia il modello ad apprendere rappresentazioni di feature che rimangono coerenti anche sotto varie alterazioni dell’immagine, inclusa la compressione tipicamente incontrata sulle piattaforme di social media. Qui, una testa di proiezione viene utilizzata per flessibilità durante l’addestramento ma alla fine scartata. L’obiettivo è che la backbone generi vettori di feature che siano resistenti a tali manipolazioni dell’immagine. Simile all’approccio supervisionato, una nuova testa viene attaccata alla backbone congelata nella terza fase. Questa testa viene poi addestrata per la classificazione binaria sul dataset TrueFace per distinguere le immagini reali da quelle false. La differenza chiave risiede nella robustezza migliorata della backbone pre-addestrata, potenzialmente riducendo la necessità di ulteriore fine-tuning specifico per ogni piattaforma di social media. Dopo la fase di fine-tuning, il modello viene valutato su immagini condivise su piattaforme come Twitter, Facebook e Telegram. Il successo dell’addestramento auto-supervisionato è misurato dalla capacità del modello di performare bene senza ulteriore fine-tuning per ogni piattaforma. Idealmente, i vettori di feature appresi durante la fase auto-supervisionata dovrebbero essere sufficientemente generalizzabili da gestire le variazioni di compressione delle immagini attraverso diversi social media.

2.2.1 SimCLR: Un Framework di Apprendimento Contrastivo

SimCLR apprende rappresentazioni massimizzando l’accordo tra versioni diversamente manipolate dello stesso esempio di dati tramite una perdita contrastiva nello spazio latente. Come illustrato nella figura 2.2, questo framework comprende i seguenti quattro componenti principali:

- Un modulo che esegue *data augmentation* stocastica su un’immagine di input. Crea due versioni modificate, \tilde{x}_i e \tilde{x}_j , dello stesso campione originale. Queste versioni modificate sono considerate



Figura 2.3: Esempi degli effetti degli operatori di augmentation. Ogni augmentation può modificare i dati stocasticamente con alcuni parametri interni (come gradi di rotazione, quantità di rumore). Una batch di 4 immagini è duplicata e augmentation diverse sono applicate su ogni immagine. Alcuni esempi di augmentation sono *specchiamento orizzontale*, *distorsione dei colori*, *Conversione in scala di grigi*, *Rumore Gaussiano*, *Sfuocamento Gaussiano*, *Rimozione di patch* and *Compressione JPEG*.

una ‘coppia positiva’ perché sono correlate tra loro pur contenendo variazioni. È fondamentale decidere attentamente l’insieme e l’intensità delle operazioni di augmentation poiché questo rende più difficile per il modello riconoscere le coppie positive. Un esempio di possibili augmentation può essere visto nella Figura 2.3. Seguendo esempi da [10], non è stato applicato alcun ridimensionamento in nessun esperimento, poiché ciò potrebbe compromettere alcuni degli artefatti lasciati da modelli come i GAN nel processo generativo.

- Un *encoder di base* di rete neurale (indicato con $f(\cdot)$) per elaborare i dati aumentati. Questo encoder può essere qualsiasi tipo di architettura di rete neurale come la popolare architettura ResNet50 [22]. L’encoder prende un punto dati aumentato (\tilde{x}_i) e lo trasforma in un vettore di rappresentazione numerica $\tilde{h}_i = f(\tilde{x}_i) = \text{ResNet}(\tilde{x}_i)$. Durante la fase di addestramento, il componente cerca di apprendere embedding di immagini che siano il più robusti possibile a varie forme di augmentation. Poiché questo componente è equivalente alla backbone dei modelli nel primo passo del metodo supervisionato, verranno testate diverse architetture.
- Un modulo MLP agisce come testa di proiezione. Questa testa mappa le rappresentazioni estratte dall’encoder di base in un nuovo spazio latente dove viene applicata la perdita contrastiva.
- Una *funzione di perdita contrastiva*, che gioca un ruolo cruciale nel compito di predizione contrastiva. Dato un insieme \tilde{x}_k che include una coppia positiva di campioni \tilde{x}_i e \tilde{x}_j , il modello mira a identificare \tilde{x}_j all’interno dell’insieme $\{\tilde{x}_k\}_{k \neq i}$ per un dato \tilde{x}_i . Essenzialmente, la perdita contrastiva incoraggia il modello ad apprendere rappresentazioni dove punti dati simili (come le versioni aumentate della stessa immagine) sono più vicini tra loro nello spazio latente.

L’algoritmo 1 riassume il metodo proposto.

La Perdita di Entropia Incrociata Normalizzata e Scalata in Temperatura

Per ottenere il risultato desiderato di avere versioni aumentate simili della stessa immagine raggruppate nello spazio latente, mentre si distanziano le rappresentazioni di immagini diverse, viene impiegata una specifica funzione di perdita. Questa funzione essenzialmente guida il processo di apprendimento del modello. Un gruppo (mini-batch) di immagini viene scelto dal dataset. Ogni immagine viene poi aumentata, creando due versioni manipolate di ciascuna immagine originale. Questo raddoppia il numero di immagini nel batch. All’interno del batch aumentato, ogni immagine agisce come un

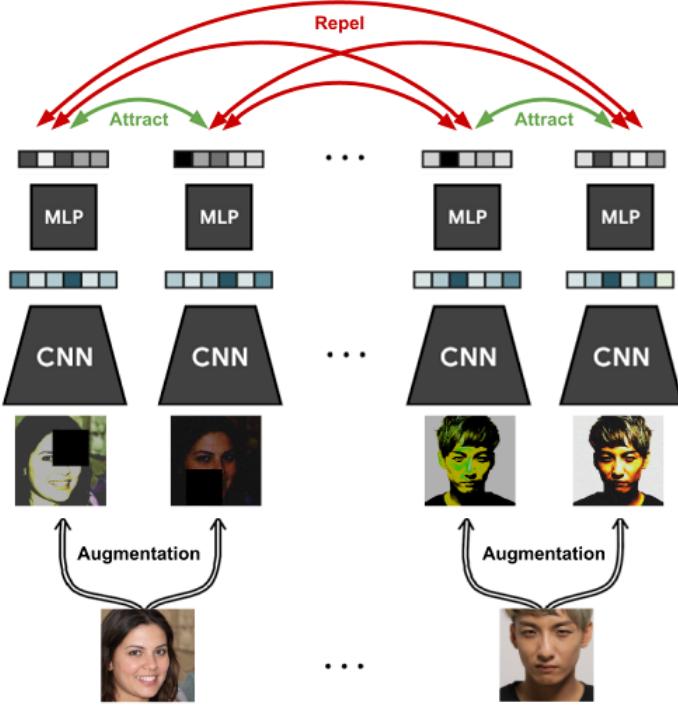


Figura 2.4: Una rappresentazione del processo di apprendimento contrastivo. Le versioni manipolate vengono passate a un encoder di base e a una testa di proiezione. Grazie alla funzione di perdita NTXent, il modello cerca di raggruppare queste versioni simili, mantenendo le rappresentazioni di immagini completamente diverse lontane tra loro.

riferimento. La sua versione aumentata corrispondente è considerata un esempio positivo (vista simile dalla stessa fonte). Tutte le immagini rimanenti (dopo aver rimosso il riferimento e la sua coppia positiva) sono trattate come esempi negativi (versioni manipolate da fonti diverse). La funzione di perdita per una coppia positiva di esempi (i, j) è definita come

$$\ell_{i,j} = -\log \frac{\exp(\text{sim}(\mathbf{z}_i, \mathbf{z}_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{k \neq i} \exp(\text{sim}(\mathbf{z}_i, \mathbf{z}_k)/\tau)} \quad (2.3)$$

dove

- $\mathbb{1}_{k \neq i} \in \{0, 1\}$ è una funzione indicatore che vale 1 se $k \neq i$
- $\text{sim}(\mathbf{z}_i, \mathbf{z}_j)$ è la similarità del coseno della proiezione di i e j
- τ denota un parametro di temperatura, che può essere passato come iperparametro.

Per analizzare approfonditamente il funzionamento di questa formula, è possibile esaminare separatamente ciascuno dei suoi componenti:

- **Copie Positive:** Ci concentriamo su coppie di immagini aumentate dalla stessa fonte (l'immagine di riferimento, i , e la sua vista corrispondente, j). Il modello mira ad apprendere rappresentazioni in cui queste coppie positive siano vicine tra loro in uno spazio latente.
- **Similarità del Coseno:** È il coseno dell'angolo tra i vettori; cioè, è il prodotto scalare dei vettori diviso per il prodotto delle loro lunghezze.

$$\text{sim}(\mathbf{z}_i, \mathbf{z}_j) = \cos \theta = \mathbf{u}^\top \mathbf{v} / \|\mathbf{u}\| \|\mathbf{v}\| \quad (2.4)$$

Ne consegue che la similarità del coseno non dipende dalle magnitudini dei vettori, ma solo dal loro angolo. La similarità del coseno appartiene sempre all'intervallo $[-1, 1]$. Ad esempio, due

Algorithm 1 SimCLR's main learning algorithm.

```

input: batch size  $N$ , constant  $\tau$ , structure of  $f$ ,  $g$ ,  $\mathcal{T}$ .
for sampled minibatch  $\{\mathbf{x}_k\}_{k=1}^N$  do
    for all  $k \in \{1, \dots, N\}$  do
        draw two augmentation functions  $t \sim \mathcal{T}$ ,  $t' \sim \mathcal{T}$ 
        # the first augmentation
         $\tilde{\mathbf{x}}_{2k-1} = t(\mathbf{x}_k)$ 
         $\mathbf{h}_{2k-1} = f(\tilde{\mathbf{x}}_{2k-1})$  # representation
         $\mathbf{z}_{2k-1} = g(\mathbf{h}_{2k-1})$  # projection
        # the second augmentation
         $\tilde{\mathbf{x}}_{2k} = t'(\mathbf{x}_k)$ 
         $\mathbf{h}_{2k} = f(\tilde{\mathbf{x}}_{2k})$  # representation
         $\mathbf{z}_{2k} = g(\mathbf{h}_{2k})$  # projection
    end for
    for all  $i \in \{1, \dots, 2N\}$  and  $j \in \{1, \dots, 2N\}$  do
         $s_{i,j} = \mathbf{z}_i^\top \mathbf{z}_j / (\|\mathbf{z}_i\| \|\mathbf{z}_j\|)$  # pairwise similarity
    end for
    define  $\ell(i, j)$  as  $\ell(i, j) = -\log \frac{\exp(s_{i,j}/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{k \neq i} \exp(s_{i,k}/\tau)}$ 
     $\mathcal{L} = \frac{1}{2N} \sum_{k=1}^N [\ell(2k-1, 2k) + \ell(2k, 2k-1)]$ 
    update networks  $f$  and  $g$  to minimize  $\mathcal{L}$ 
end for
return encoder network  $f(\cdot)$ , and throw away  $g(\cdot)$ 

```

vettori proporzionali hanno una similarità del coseno di 1, due vettori ortogonali hanno una similarità di 0, e due vettori opposti hanno una similarità di -1 . È importante notare che $\mathbb{1}_{k \neq i}$ viene usato nella formula perché il punteggio di similarità di un'immagine con se stessa sarà sempre $\text{sim}(\mathbf{z}_i, \mathbf{z}_i) = 1$.

- Softmax e Temperatura: La formula incorpora una funzione softmax con un parametro di temperatura (τ). Softmax trasforma i punteggi di similarità in una distribuzione di probabilità, indicando quanto è probabile che il modello identifichi l'immagine target corretta (coppia positiva) rispetto a tutte le altre opzioni nel batch.

$$\sigma(\mathbf{z})_j = \frac{\exp(z_j)}{\sum_{k=1}^K \exp(z_k)}, \forall j = 1, \dots, K \quad (2.5)$$

La temperatura controlla quanto è enfatizzata sui picchi questa distribuzione di probabilità. Una bassa temperatura porta a previsioni più sicure, mentre una temperatura più alta incoraggia previsioni più morbide con meno certezza.

Poiché il denominatore dell'Eq. 2.3 dipende dal numero di elementi nel batch, è interessante osservare che la dimensione del batch ha una forte influenza sul valore della perdita. Questo può essere dimostrato studiando quale sarebbe il valore della perdita per un modello casuale. In particolare, supponendo una dimensione del batch di N , il modello dovrà trovare l'immagine di riferimento tra $2N - 1$ immagini. Supponendo che il modello sia perfettamente bilanciato e assegni una probabilità di $1/(2N - 1)$ a ciascuna immagine, allora la perdita sarebbe $-\log(1/(2N - 1)) = \log(2N - 1)$. Esempi possono essere visti nella tabella 2.5.

Abbiamo visto che l'approccio implica l'uso di due versioni manipolate della stessa immagine e sfrutta l'apprendimento contrastivo per codificare queste versioni in modo simile. È interessante osservare le conseguenze dell'uso di 3 o più versioni mentre si cerca di applicare lo stesso principio. La differenza dall'approccio standard con due versioni manipolate sarebbe sostanziale in termini di formulazione del problema. Per ogni immagine, ci sarebbero più di un'immagine corrispondente, trasformando il problema da etichetta singola a etichetta multipla. Di conseguenza, la funzione di perdita richiederebbe ulteriori adattamenti.

Batch size	Random Guessing Loss
2	1.09
4	1.94
8	2.70
16	3.43
32	4.14

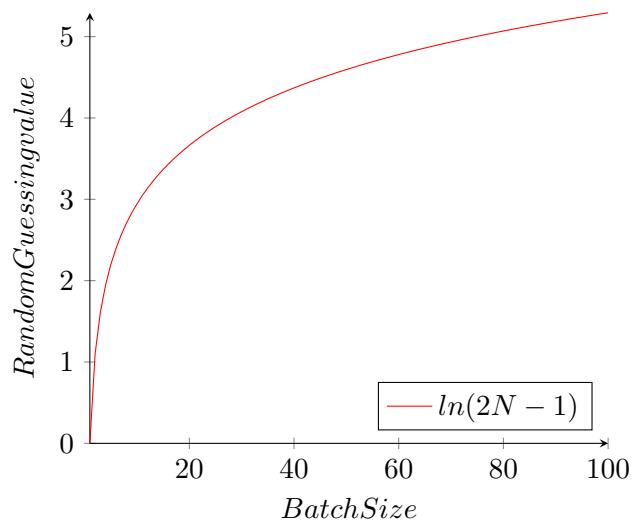


Figura 2.5: Esempi e valori per gli effetti sulla dimensione della batch sul valore della funzione di perdita con un modello perfettamente casuale

3 Setup Sperimentale

In questo capitolo, descriviamo in dettaglio il setup sperimentale impiegato per valutare l'efficacia degli approcci e dei modelli proposti per il rilevamento dei deepfake. Ciò include una descrizione del dataset utilizzato per l'addestramento e il test, l'architettura specifica dei modelli di IA, le metriche di valutazione selezionate e gli iperparametri scelti per riprodurre i risultati.

3.1 Dataset

Una parte centrale di questo lavoro enfatizza lo sforzo di affrontare il problema di portare il rilevamento dei deepfake in uno scenario reale, dove le immagini vengono caricate su diversi social media e l'operazione di compressione compromette gli artefatti utili per i modelli. Per questo motivo, è stato scelto il dataset *TrueFace*. Infatti, non solo contiene immagini reali di volti da FFHQ e immagini false da StyleGAN e StyleGAN2, ma fornisce anche una sezione post-social, dove copie di queste immagini sono state caricate e scaricate da Twitter, Telegram e Facebook. Ciò consente esperimenti e confronti analizzando gli effetti della compressione dei media sulle capacità di rilevamento di un modello. Il dataset TrueFace è il primo database pubblicamente disponibile per la classificazione di immagini sintetiche vs reali con dati condivisi anche tramite social network. Il dataset *pre-social* contiene un totale di 150K immagini di volti, di cui 70K sono reali e 80K sono sintetiche. Quelle sintetiche sono state generate da due popolari modelli generatori, ovvero StyleGAN [27] e StyleGAN2 [30], 40K ciascuno. Inoltre, le immagini per ciascun modello sono state equamente divise secondo il parametro di fantasia di StyleGAN (20K per parametro). Il parametro di fantasia ψ determina quanto lontano la rete generativa dovrebbe deviare dalla media dei dati. Regolando questo parametro, è possibile determinare il compromesso tra varietà e qualità dei dati di output. Le immagini reali sono state ottenute dal Dataset FFHQ [27], originariamente proposto come benchmark per le GAN. Sia le immagini reali che quelle sintetiche sono in formato RGB e hanno una risoluzione di 1024×1024 pixel. La seguente tabella riassume il contenuto del dataset pre-social.

Real	StyleGAN		StyleGAN2	
	$\psi = 0.5$	$\psi = 0.7$	$\psi = 0.5$	$\psi = 1$
70K	20K	20K	20K	20K

Tabella 3.1: Pre-social dataset.

Il dataset post-social contiene in totale 60K immagini, metà reali e metà false, caricate e scaricate su tre delle piattaforme di social media più popolari: Facebook (FB), Telegram (TG), Twitter (TW). Solo una frazione di 20K immagini nel dataset post-social per ogni social (10K reali, 5K da StyleGAN e 5K da StyleGAN2) proviene dal dataset pre-social. Tuttavia, le immagini nelle tre sezioni sono le stesse, con l'unica differenza che è l'operazione di compressione.

Platform	Real	StyleGAN	StyleGAN2
<i>Facebook</i> (FB)	10k	5k	5k
<i>Telegram</i> (TG)	10k	5k	5k
<i>Twitter</i> (TW)	10k	5k	5k

Tabella 3.2: Post-social dataset

Al fine di presentare risultati comparabili, il set di test è composto dalle stesse immagini sia negli scenari pre che post compressione dei media. Ad esempio, il set di test sulla sezione Twitter è lo stesso

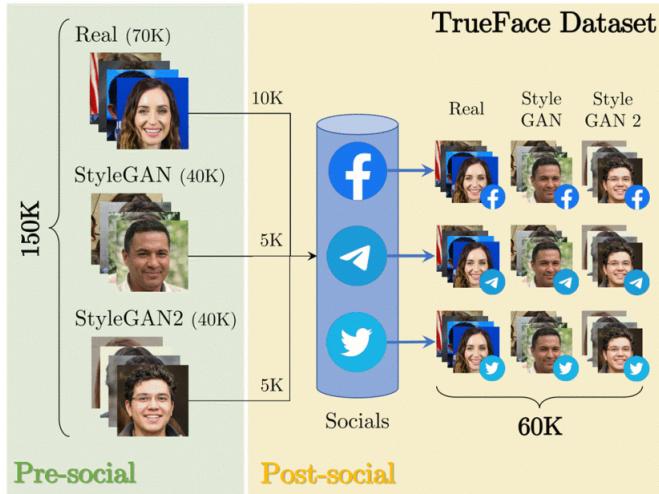


Figura 3.1: Il dataset TrueFace. La collezione pre-social include 70k volti reali e 80k immagini generate da GAN, metà da StyleGAN [27] e metà da StyleGAN2 [30]. La collezione post-social include la versione condivisa di una parte della sezione pre-social, dove le immagini sono state caricate e scaricate da Facebook, Telegram e Twitter, per un totale di 60k immagini. Il dataset è disponibile su <https://bit.Ly/3baeh75>.

del set di test sulle immagini pre-social, una volta che è stato caricato sul social media. Questo viene fatto per ottenere risultati comparabili tra diversi approcci e test.

3.2 Architetture

Questo lavoro analizza gli approcci su tre diverse architetture. In primo luogo, viene utilizzato una ResNet50 standard [22]. Lo strato completamente connesso alla fine viene rimosso e il metodo forward si basa sull'estrazione delle caratteristiche e sullo strato di pooling medio. I pesi pre-addestrati su ImageNet possono essere trovati sulla pagina web di NVIDIA¹. L'architettura PhiNets [40] è stata utilizzata anche come backbone come mostrato nella Figura 1.7. Seguendo l'implementazione² e i pesi sul sito web di Hugging Face³, questi sono i parametri di costruzione per la rete: $\alpha = 1.1$, $\beta = 0.75$, $t_0 = 5$, $N = 8$. L'implementazione di MicroNet [33] e i pesi pre-addestrati sono stati presi dalla repository ufficiale⁴. Tra le diverse opzioni disponibili, è stata scelta la versione MicroNet-M1 con 1,8 milioni di parametri, in quanto questa è la versione con una quantità di parametri comparabile all'architettura di PhiNet.

3.3 Apprendimento Supervisionato

Nell'approccio supervisionato, ogni architettura utilizza il proprio backbone congelato e sfrutta una testa per classificare la codifica del vettore di caratteristiche. Mentre ResNet50 viene fornito con un vettore di caratteristiche su 2048 dimensioni, Phinet e MicroNet vengono forniti con un vettore più piccolo di circa 200 elementi. Per questo motivo, mentre ResNet utilizza un singolo strato lineare di forma [2048, 1], la testa delle altre architetture utilizza anche uno strato nascosto di dimensione 512 per aumentare l'espressività del vettore di caratteristiche. L'addestramento è stato condotto per un totale di 10 epoch. Il tasso di apprendimento è impostato a $2e - 5$ e la dimensione del batch è 4.

3.4 Apprendimento Auto-Supervisionato

Per il compito pretesto, la scelta delle operazioni di aumento da eseguire è cruciale. In particolare, è importante evitare lo shearing e il ridimensionamento, poiché ciò compromette gli artefatti

¹ResNet50 pre-addestrato su ImageNet: https://catalog.ngc.nvidia.com/orgs/nvidia/models/resnet50_pytorch_amp

²Repository Micromind con codice PhiNet: <https://github.com/micromind-toolkit/micromind>

³Tutorial Hugging Face per PhiNet: <https://huggingface.co/micromind/ImageNet>

⁴Implementazione e pesi di MicroNet: <https://github.com/liyunsheng13/micronet/tree/main>

che il modello deve distinguere [10]. Seguendo gli esempi di [10, 7], il modulo di augmentation è una composizione di *Flip Orizzontale*, *Distorsione del Colore (jitter)*, *Sfocatura Gaussiana*, *Rumore Gaussiano*, *Conversione in Scala di Grigi*, *Cut Out* e *Compressione JPEG*. Sequenzialmente, le immagini vengono anche normalizzate con valori standard per media e deviazione standard (media = [0.485, 0.456, 0.406], deviazione standard = [0.229, 0.224, 0.225]). La Figura 2.3 mostra alcuni risultati. La dimensione del batch è due, il che significa che ad ogni passo, data un riferimento, il modello mira a trovare la coppia positiva tra altre tre immagini (una è corretta e due non lo sono). Il valore della temperatura è impostato a 0,07, che è anche il valore predefinito. Questa è considerata una temperatura bassa che enfatizza piccole preferenze del modello per un'immagine rispetto a un'altra. La testa di proiezione è composta da un MLP completamente connesso con uno strato nascosto di dimensione 2048, una funzione di attivazione ReLU e un altro strato con una dimensione di output di 128, che è il numero di dimensioni dello spazio latente. L'addestramento è condotto con un tasso di apprendimento fisso di $2e - 5$ per tutti i modelli. Per la fase supervisionata, solo la normalizzazione viene eseguita nella fase di preprocessamento. Il tasso di apprendimento viene mantenuto costante e la dimensione del batch è quattro. Dopo la fase di apprendimento contrastivo, il backbone del modello viene congelato e la successiva fase supervisionata consiste solo nell'addestrare una nuova testa con la stessa struttura di prima (con uno strato nascosto per le architetture di bordo).

3.5 Metriche

In questa sezione, illustriamo le metriche di valutazione impiegate per valutare le prestazioni dei modelli. Data la natura del nostro compito, abbiamo selezionato una combinazione di metriche che forniscono una visione completa dell'efficacia del modello. Queste metriche includono l'accuratezza, l'Area Sotto la Curva ROC (Area Under Curve - AUC) e il punteggio F1. Discutiamo la logica alla base della scelta di queste metriche e il loro significato nel contesto dei nostri specifici obiettivi. Uno strumento fondamentale per comprendere le prestazioni del modello di classificazione è la matrice di confusione, che riassume i conteggi dei veri positivi (True Positives - TP), veri negativi (True Negatives - TN), falsi positivi (False Positives - FP) e falsi negativi (False Negatives - FN). Ai fini di questo lavoro, positivo è inteso come Reale e negativo come Falso, quindi i falsi positivi indicano la quantità di immagini False che il modello ha valutato come Reali.

Accuratezza L'accuratezza rappresenta la proporzione di previsioni corrette fatte dal modello su tutte le istanze. Matematicamente, è calcolata come

$$\text{Accuratezza} = \frac{TP + TN}{TP + FP + TN + FN} \quad (3.1)$$

Mentre l'accuratezza è una metrica semplice e intuitiva, può essere fuorviante in situazioni con distribuzioni di classe sbilanciate, dove una classe è significativamente più prevalente dell'altra. In tali casi, un'alta accuratezza potrebbe non riflettere necessariamente buone prestazioni sulla classe minoritaria. Sebbene nel nostro caso il dataset sia solo leggermente sbilanciato (46% Reale, 54% Falso), integriamo l'accuratezza con metriche aggiuntive come AUC e punteggio F1, che forniscono una comprensione più sfumata delle prestazioni del modello tra le diverse classi.

Punteggio F1 Il punteggio F1 è una metrica preziosa che combina precisione e richiamo in un unico valore, fornendo una valutazione equilibrata delle prestazioni di un modello. La precisione misura la proporzione di previsioni vere positive su tutte le previsioni positive fatte dal modello. In altre parole, risponde alla domanda: "Di tutte le istanze previste come positive (Reali), quante erano effettivamente positive?". È descritta come:

$$\text{Precisione} = \frac{TP}{TP + FP} \quad (3.2)$$

Il Richiamo (noto anche come sensibilità) misura la proporzione di previsioni vere positive su tutte le istanze positive effettive. Risponde alla domanda: "Di tutte le istanze positive (Reali) effettive,

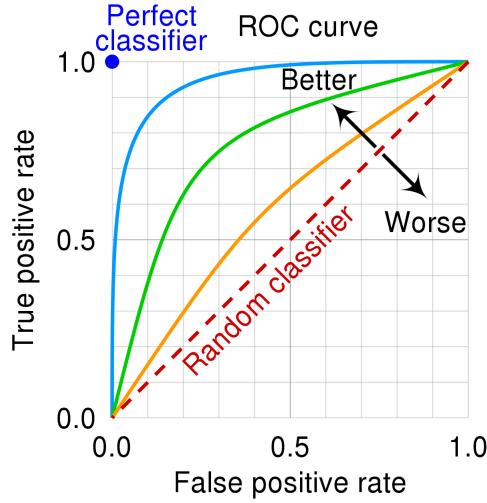


Figura 3.2: Lo spazio ROC per un classificatore "migliore" e "peggiore".

quante ne ha correttamente identificate il modello?" Il richiamo è calcolato come:

$$\text{Richiamo} = \frac{TP}{TP + FN} \quad (3.3)$$

Il punteggio F1 è la media armonica di precisione e richiamo, dando uguale peso a entrambi. È calcolato come:

$$\text{Punteggio F1} = \frac{2 \times \text{Precisione} \times \text{Richiamo}}{\text{Precisione} + \text{Richiamo}} \quad (3.4)$$

Un punteggio F1 più alto indica migliori prestazioni del modello, con un punteggio F1 perfetto di 1 raggiunto quando sia la precisione che il richiamo sono perfetti.

AUC AUC è l'acronimo di Area Under Curve (Area Sotto la Curva). Tale curva è la curva Receiver Operating Characteristic (ROC), ed è un grafico che illustra la capacità diagnostica di un sistema di classificazione binaria al variare della sua soglia di discriminazione. Rappresenta graficamente il Richiamo, noto anche come True Positive Rate (TPR), rispetto al False Positive Rate (FPR). L'FPR è definito come

$$\text{FPR} = \frac{FP}{FP + TN} \quad (3.5)$$

ed è la probabilità che un'immagine Falsa passi attraverso il filtro e venga valutata come Reale.

La Figura 3.2 mostra un esempio di come vengono tracciate le curve ROC e qual è il loro significato.

L'AUC è l'area sotto questa curva ROC. Un modello con perfetta discriminazione tra classi positive e negative avrà un AUC di 1, mentre un modello senza capacità di discriminazione avrà un AUC di 0,5 (equivalente a indovinare a caso).

4 Risultati

In questo capitolo, analizzeremo i risultati ottenuti dagli approcci precedentemente impiegati nelle tre diverse architetture e valuteremo l'efficacia dell'utilizzo dell'Apprendimento Contrastivo per questo specifico compito.

Approccio supervisionato Il primo metodo testato è l'approccio supervisionato, in cui il modello viene addestrato prima su immagini pre-social e successivamente ottimizzato su immagini compresse. Le Tabelle 4.1, 4.2 e 4.3 mostrano la grande capacità delle reti di sfruttare lo feature vector prodotto dal backbone per classificare correttamente le immagini. È importante ricordare che il vettore di caratteristiche è il risultato del pre-addestramento del modello sul dataset ImageNet, che conferisce alla rete competenze in termini di interpretazione generale di un'immagine. I modelli non possono performare altrettanto bene su immagini condivise sui social media. Secondo i risultati, c'è un forte calo delle prestazioni senza fine-tuning, anche se non emerge un chiaro schema tra i diversi social media. Come era stato anticipato nelle sezioni 1.1.2 e 1.2, gli artefatti invisibili nascosti nell'immagine sono molto delicati e vengono alterati dal processo di caricamento, portando agli effetti evidenti. Tuttavia, è possibile colmare il divario ottimizzando il modello su un social specifico. Lo svantaggio è che ora viene prodotta una testa diversa quando il modello viene ottimizzato su ciascun social, così da avere una testa per Telegram, una per Facebook e una per Twitter. Nell'articolo TrueFace [3] si mostra che una testa per Twitter ha anche alcune capacità per Telegram e Facebook e viceversa, ma con prestazioni inferiori. Inoltre, è importante notare come le prestazioni variano tra le diverse architetture. Sebbene le differenze di dimensioni tra queste reti verranno discusse in seguito, vale la pena notare che ResNet50 è quella con i risultati più alti in media, ma PhiNet e MicroNet ottengono buoni risultati. MicroNet è quasi allo stesso livello di ResNet50 sulle immagini pre-Social, mentre PhiNet si attesta su prestazioni inferiori. Tuttavia, quest'ultimo sembra più robusto su immagini post-social senza fine-tuning. Dopo l'addestramento su immagini post-social, ResNet50 è quello che si avvicina di più ai risultati pre-social, seguito da MicroNet e PhiNet.

Accuracy	PRE SOCIAL	POST SOCIAL					
		No Fine Tuning			Fine Tuning		
		Facebook	Telegram	Twitter	Facebook	Telegram	Twitter
ResNet50	99.9%	54.6%	59.5%	78.3%	93.1%	97.0%	94.6%
PhiNet	93.1%	66.8%	74.2%	57.2%	85.0%	83.6%	91.5%
MicroNet	99.8%	46.6%	46.6%	46.6%	86.6%	93.6%	86.6%

Tabella 4.1: Risultati di accuratezza con l'apprendimento supervisionato

F1 Score	PRE SOCIAL	POST SOCIAL					
		No Fine Tuning (F1 Score)			Fine Tuning (F1 Score)		
		Facebook	Telegram	Twitter	Facebook	Telegram	Twitter
ResNet50	92.5%	63.1%	72.3%	65.4%	85.3%	88.6%	87.3%
PhiNet	85.4%	67.2%	72.3%	63.4%	73.0%	79.6%	84.3%
MicroNet	91.5%	59.4%	59.1%	59.5%	77.2%	86.1%	80.6%

Tabella 4.2: Punteggio f1 con apprendimento supervisionato

	AUC	PRE SOCIAL	POST SOCIAL					
			No Fine Tuning (AUC)			Fine Tuning (AUC)		
			Facebook	Telegram	Twitter	Facebook	Telegram	Twitter
ResNet50	87.6%		64.8%	83.9%	61.7%	85.9%	86.0%	85.8%
PhiNet	85.3%		68.5%	78.0%	54.6%	81.1%	83.8%	85.3%
MicroNet	86.8%		46.3%	72.4%	54.5%	79.5%	86.1%	86.9%

Tabella 4.3: Risultati in termini di AUC con apprendimento supervisionato

Approccio auto-supervisionato Il secondo approccio sfrutta l'apprendimento auto-supervisionato, con un pretext task basato sull'apprendimento contrastivo per ottenere robustezza nelle operazioni di compressione dei media e una fase supervisionata per il rilevamento di Deepfake. Pertanto, i seguenti risultati presentano le prestazioni dei modelli dopo che hanno attraversato sia il pretext task che la fase supervisionata. I risultati pre-social mostrano che le prestazioni presentano un divario rispetto all'approccio standard. Ciò significa che dopo la fase preliminare, il backbone del modello apprende caratteristiche che non sono abbastanza dettagliate per distinguere efficacemente tra immagini reali e false. D'altra parte, gli effetti dell'Apprendimento Contrastivo sono apprezzabili. Il calo tra le prestazioni su immagini pre e post-social è quasi completamente trascurabile e in alcuni casi i risultati post-social sono addirittura migliori (Tabella 4.4). Questo è particolarmente interessante, considerando che il modello è stato addestrato solo su immagini pre-social. Questa coerenza tra i risultati pre e post-social potrebbe suggerire che, aumentando ora le prestazioni nel pre-social, seguirà un miglioramento nel post-social. Dopo alcuni tentativi, ci siamo accorti che non è un risultato semplice da ottenere. Oltre a variare gli iperparametri e l'intensità dell'augmentation, il tentativo più vicino è stato quello di evitare di congelare il backbone dopo il pretext task. La speranza è di prendere le rappresentazioni stabili dopo SimCLR e ottimizzare il backbone per fargli produrre caratteristiche specifiche per il rilevamento di Deepfake. Effettivamente, facendo questo si ottengono alte prestazioni nella fase supervisionata, simili ai risultati pre-social nella tabella 4.1, ma segue lo stesso schema quando testato su immagini post-social. Scongelare il backbone permette al modello di "dimenticare" la robustezza raggiunta precedentemente e presentare lo stesso calo di accuratezza misurato prima. Questi risultati ricordano in qualche modo il compromesso Stabilità / Plasticità nell'Apprendimento Continuo. Per imparare un nuovo compito, cioè rendere le caratteristiche di ImageNet robuste all'augmentation, il modello dimentica parzialmente le vecchie caratteristiche e non può performare altrettanto bene quando addestrato sul compito Reale vs Falso. Analogamente a prima, le architetture edge non stanno performando male rispetto a ResNet50, anche se c'è un divario tra quest'ultima e PhiNet. D'altra parte, i numeri mostrano che MicroNet è quella con i risultati più alti in termini di accuratezza, punteggio f1 e area sotto la curva, indicando che una riduzione dei requisiti di potenza e del numero di parametri non sta compromettendo affatto le prestazioni.

Accuracy	PRE SOCIAL	POST SOCIAL		
		Facebook	Telegram	Twitter
ResNet50	67.1%	66.8%	67.5%	67.1%
PhiNet	63.8%	61.2%	64.8%	63.3%
MicroNet	70.5%	63.8%	70.1%	69.7%

Tabella 4.4: Risultati di accuratezza con l'apprendimento auto-supervisionato

Dimensione del modello Come notato in precedenza, PhiNet e MicroNet hanno avuto prestazioni competitive con ResNet50 in entrambi gli approcci. Tuttavia, per apprezzare pienamente le loro prestazioni, è cruciale considerare le significative differenze nella dimensione del modello. La Tabella 4.7 rivela che ResNet50 ha 23 milioni di parametri, mentre PhiNet e MicroNet hanno circa un ventesimo di quella dimensione. La disparità è ancora più sorprendente quando si considera il numero di

AUC	PRE SOCIAL	POST SOCIAL		
		Facebook	Telegram	Twitter
ResNet50	72.8%	61.3%	65.6%	63.6%
PhiNet	68.3%	58.4%	61.9%	59.3%
MicroNet	63.8%	57.8%	65.3%	63.9%

Tabella 4.5: Risultati in termini di AUC con apprendimento auto-supervisionato

F1 Score	PRE SOCIAL	POST SOCIAL		
		Facebook	Telegram	Twitter
ResNet50	59.3%	51.5%	54.3%	54.4%
PhiNet	62.0%	46.1%	56.3%	54.6%
MicroNet	61.6%	44.1%	62.4%	61.0%

Tabella 4.6: Punteggio f1 con apprendimento auto-supervisionato

operazioni. L’analisi di un’immagine 1024 x 1024 richiede 85 miliardi di operazioni per ResNet50, ma solo 4 miliardi per PhiNet e appena 7,6 milioni (<1% di ResNet) per MicroNet. Questa sostanziale differenza tra PhiNet e MicroNet nasce perché MicroNet è specificamente progettato per ottimizzare il numero di operazioni, come dettagliato nella sezione 1.3.3. Al contrario, PhiNet dà priorità alla scalabilità per soddisfare specifici requisiti hardware. Mentre la tecnica di micro-fattorizzazione utilizzata in MicroNet (discussa nella sezione 1.3.3 e [33]) potrebbe potenzialmente essere applicata a PhiNet, questa strada non è ancora stata esplorata.

I nostri risultati dimostrano un chiaro compromesso tra la riduzione dei parametri e delle operazioni (e quindi del consumo energetico) e le prestazioni complessive. Tuttavia, a seconda del caso d’uso specifico, i vantaggi delle architetture edge, come una riduzione del 99,9% delle operazioni, possono superare il minore calo di prestazioni dello 0,1MicroNet ha dimostrato un’eccezionale efficacia nel compito Reale vs Falso, in particolare in ambienti controllati. Questi risultati supportano fortemente l’uso di tali reti per compiti computazionalmente intensivi. Data l’impressionante accuratezza raggiunta nei metodi supervisionati, è concepibile che uno scenario futuro possa coinvolgere un’architettura edge come MicroNet che pre-valida i post sui dispositivi prima che vengano caricati sui social media.

	MACC	Number of parameters
ResNet50	85.4 B	23.5 M
Phinet	4.07 B	1.14 M
Micronet	7.62 M	1.44 M

Tabella 4.7: Numero di operazioni e parametri per ogni modello. Il numero di operazioni dipende dalla risoluzione dell’immagine

Conclusioni

La proliferazione dei deepfake solleva significative preoccupazioni etiche, in particolare riguardo al loro potenziale di compromettere la fiducia nei media digitali, manipolare l’opinione pubblica e perpetuare stereotipi o pregiudizi dannosi. Con l’evolversi della tecnologia per la generazione di deepfake, che diventa sempre più sofisticata e accessibile, i rischi di un uso malevolo aumentano, minacciando di minare la credibilità dei contenuti autentici e di esacerbare le divisioni sociali esistenti. Inoltre, il potenziale utilizzo dei deepfake nel cyberbullismo, nelle molestie e in altre forme di attacco personale solleva serie preoccupazioni sulla privacy e la sicurezza individuale. Affrontare queste sfide etiche richiede un approccio multiforme che coinvolga progressi tecnologici nella rilevazione e mitigazione, nonché più ampie discussioni sociali sulla alfabetizzazione mediatica e l’uso responsabile dei contenuti generati dall’IA.

Negli ultimi anni, il rilevamento dei deepfake è diventato sempre più critico a causa dei progressi nei modelli generativi come StyleGAN e Diffusion Models, che producono media sintetici visivamente indistinguibili dai contenuti reali. Molte ricerche si concentrano sullo sviluppo di modelli robusti che generalizzino su varie architetture generative sfruttando gli artefatti introdotti durante il processo di creazione. Tuttavia, questi artefatti sono spesso significativamente alterati quando i media vengono caricati e condivisi online, subendo compressione e ridimensionamento.

Questa tesi indaga sui compromessi che emergono quando i modelli di rilevamento dei deepfake devono performare in modo coerente sia su immagini pristine che compresse, uno scenario comune negli ambienti dei social media del mondo reale. Sono state esplorate due strategie principali: l’uso dell’apprendimento auto-supervisionato per ottenere robustezza rispetto alla compressione dei social media, e l’uso di architetture edge per esplorare il potenziale di futuri controlli di validità on-device delle immagini prima del caricamento.

Abbiamo esaminato il potenziale dell’SSL per migliorare la robustezza del modello agli artefatti di compressione dei social media, mirando specificamente a piattaforme come Telegram, Facebook e Twitter. Ciò viene fatto sfruttando SimCLR, un framework di apprendimento contrastivo proposto da Google, che mira a insegnare al modello a distinguere tra coppie di immagini simili (versioni manipolate della stessa immagine) e dissimili. Facendo ciò prima della fase di apprendimento supervisionato sul compito Reale vs Falso, i modelli possono catturare le caratteristiche essenziali delle immagini e diventare meno sensibili alle variazioni causate dalla compressione. La nostra ricerca dimostra che l’SSL può efficacemente colmare il divario di prestazioni tra contenuti pre e post social media. Tuttavia, questa robustezza viene a scapito dell’accuratezza complessiva, suggerendo che un approccio supervisionato potrebbe essere preferibile in situazioni in cui l’accuratezza è di primaria importanza.

Inoltre, abbiamo valutato le prestazioni di reti neurali leggere (MicroNet e PhiNet) per il rilevamento dei deepfake on-device, mirando a consentire un controllo di validità in tempo reale. I nostri risultati rivelano che, in ambienti controllati, le MicroNet possono egualizzare l’accuratezza di modelli più grandi come ResNet50 riducendo significativamente la complessità computazionale. L’accuratezza del 99,8% di MicroNet su immagini pre-social è un risultato sorprendente. Sebbene i risultati post-social per MicroNet non siano ancora così vicini ai modelli più grandi, i risultati pre-social aprono possibilità per un efficiente rilevamento dei deepfake direttamente sui dispositivi degli utenti prima del caricamento.

La ricerca futura può estendere questo lavoro in molteplici direzioni. Poiché l’Apprendimento Supervisionato si è dimostrato quello che fornisce una maggiore accuratezza sulle immagini post-social, è interessante esplorare ulteriormente il suo utilizzo. Ad esempio, potrebbe valere la pena analizzare un’immagine e cercare di capire quali social network hanno operato su quell’immagine in precedenza, per utilizzare una testa di classificazione specificamente addestrata per quello specifico social media. Questo aiuterebbe a risolvere il problema analizzato nella sezione 4 di avere diverse teste di classificazione, ognuna specializzata su un certo social. Tuttavia, ciò richiederebbe un’ulteriore elaborazione

che deve essere ottimizzata per permettere ai dispositivi edge di sostenere il carico computazionale. In alternativa, approcci di questo tipo potrebbero essere analizzati anche su diverse compressioni dei social media e modelli generativi come Diffusion Models [12] o StyleGAN3 [29]. Con l’evolversi continuo della tecnologia deepfake, è cruciale affrontare le implicazioni etiche e aumentare la consapevolezza pubblica sui potenziali rischi e danni associati ai deepfake. La ricerca futura potrebbe concentrarsi sullo sviluppo di risorse educative e strumenti per aiutare gli individui a identificare e valutare criticamente i deepfake, vengono distribuiti online.

Bibliografia

- [1] Alberto Ancilotto, Francesco Paissan, and Elisabetta Farella. Xinet: Efficient neural networks for tinyml. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 16922–16931, 2023.
- [2] Ben Athiwaratkun and Keegan Kang. Feature representation in convolutional neural networks, 2015.
- [3] Giulia Boato, Cecilia Pasquini, Antonio L. Stefani, Sebastiano Verde, and Daniele Miorandi. Trueface: a dataset for the detection of synthetic face images from social networks. In *2022 IEEE International Joint Conference on Biometrics (IJCB)*, pages 1–7, 2022.
- [4] Sergi D Bray, Shane D Johnson, and Bennett Kleinberg. "testing human ability to detect 'deepfake' images of human faces". *Journal of Cybersecurity*, 9(1):tyad011, 06 2023.
- [5] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2019.
- [6] L. Chai, D. Bau, S.-N. Lim, and P. Isola. What makes fake images detectable? Understanding properties that generalize. In *ECCV*, 2020.
- [7] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations, 2020.
- [8] Riccardo Corvi, Davide Cozzolino, Giada Zingarini, Giovanni Poggi, Koki Nagano, and Luisa Verdoliva. On the detection of synthetic images generated by diffusion models, 2022.
- [9] D. Cozzolino, J. Thies, A. Rössler, C. Riess, M. Nießner, and L. Verdoliva. ForensicTransfer: Weakly-supervised domain adaptation for forgery detection. *arXiv preprint arXiv:1812.02510*, 2018.
- [10] Davide Cozzolino, Diego Gragnaniello, Giovanni Poggi, and Luisa Verdoliva. Towards universal gan image detection, 2021.
- [11] B. Dayma, S. Patil, P. Cuenca, K. Saifullah, T. Abraham, P. Lê Khắc, L. Melas, and R. Ghosh. DALL-E Mini, 7 2021.
- [12] P. Dhariwal and A. Nichol. Diffusion models beat GANs on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021.
- [13] M. Du, S. Pentyala, Y. Li, and X. Hu. Towards generalizable forgery detection with locality-aware autoencoder. *ACM CIKM*, 2019.
- [14] R. Durall, M. Keuper, and J. Keuper. Watch your up-convolution: CNN based Generative Deep Neural Networks are failing to reproduce spectral distributions. In *CVPR*, 2020.
- [15] Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis, 2021.
- [16] L. Nataraj et al. Detecting GAN generated fake images using co-occurrence matrices. In *IS&T EI, Media Watermarking, Security, and Forensics*, 2019.

- [17] J. Frank, T. Eisenhofer, L. Schönherr, A. Fischer, D. Kolossa, and T. Holz. Leveraging Frequency Analysis for Deep Fake Image Recognition. In *CVPR*, 2020.
- [18] Ian J. Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks, 2015.
- [19] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.
- [20] Diego Gragnaniello, Davide Cozzolino, Francesco Marra, Giovanni Poggi, and Luisa Verdoliva. Are gan generated images easy to detect? a critical analysis of the state-of-the-art, 2021.
- [21] Hui Guo, Shu Hu, Xin Wang, Ming-Ching Chang, and Siwei Lyu. Eyes tell all: Irregular pupil shapes reveal gan-generated faces, 2022.
- [22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.
- [23] Mi-Young Huh, Pulkit Agrawal, and Alexei A. Efros. What makes imagenet good for transfer learning? *CoRR*, abs/1608.08614, 2016.
- [24] Rakhee Kallimani, Krishna Pai, Prasoon Raghuwanshi, Sridhar Iyer, and Onel L. A. López. Tinyml: Tools, applications, challenges, and future research directions. *Multimedia Tools and Applications*, 83(10):29015–29045, September 2023.
- [25] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation, 2018.
- [26] Tero Karras, Miika Aittala, Samuli Laine, Erik Häkkinen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-free generative adversarial networks, 2021.
- [27] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. *CoRR*, abs/1812.04948, 2018.
- [28] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks, 2019.
- [29] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. *CoRR*, abs/1912.04958, 2019.
- [30] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan, 2020.
- [31] Federica Lago, Cecilia Pasquini, Rainer Böhme, Hélène Dumont, Valérie Goffaux, and Giulia Boato. More real than real: A study on human visual perception of synthetic faces. *CoRR*, abs/2106.07226, 2021.
- [32] H. Li, B. Li, S. Tan, and J. Huang. Detection of deep network generated images using disparities in color components. *Signal Processing*, 174, 2020.
- [33] Yunsheng Li, Yinpeng Chen, Xiyang Dai, Dongdong Chen, Mengchen Liu, Lu Yuan, Zicheng Liu, Lei Zhang, and Nuno Vasconcelos. Micronet: Improving image recognition with extremely low flops, 2021.
- [34] Alexandra Sasha Luccioni, Sylvain Viguer, and Anne-Laure Ligozat. Estimating the carbon footprint of bloom, a 176b parameter language model, 2022.
- [35] F. Marra, C. Saltori, G. Boato, and L. Verdoliva. Incremental learning for the detection and classification of gan-generated images. In *IEEE WIFS*, 2019.

- [36] Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. volume 24 of *Psychology of Learning and Motivation*, pages 109–165. Academic Press, 1989.
- [37] S. McCloskey and M. Albright. Detecting GAN-Generated Imagery Using Saturation Cues. In *IEEE ICIP*, 2019.
- [38] Thanh Thi Nguyen, Quoc Viet Hung Nguyen, Dung Tien Nguyen, Duc Thanh Nguyen, Thien Huynh-The, Saeid Nahavandi, Thanh Tam Nguyen, Quoc-Viet Pham, and Cuong M. Nguyen. Deep learning for deepfakes creation and detection: A survey. *Computer Vision and Image Understanding*, 223:103525, October 2022.
- [39] A. Nichol, P. Dhariwal, A. Ramesh, P. Shyam, P. Mishkin, B. McGrew, I. Sutskever, and M. Chen. GLIDE: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741*, 2021.
- [40] Francesco Paissan, Alberto Ancilotto, and Elisabetta Farella. Phinets: a scalable backbone for low-power ai at the edge, 2021.
- [41] Francesco Paissan, Alberto Ancilotto, and Elisabetta Farella. Phinets: A scalable backbone for low-power ai at the edge. *ACM Trans. Embed. Comput. Syst.*, 21(5), dec 2022.
- [42] David Patterson, Joseph Gonzalez, Urs Hözle, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. The carbon footprint of machine learning training will plateau, then shrink, 2022.
- [43] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125v1*, 2022.
- [44] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, pages 10684–10695, 2022.
- [45] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. Stable diffusion. <https://github.com/CompVis/stable-diffusion>, 2022.
- [46] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks, 2019.
- [47] L. Verdoliva. Media forensics and deepfakes: an overview. *IEEE Journal of Selected Topics in Signal Processing*, 14(5):910–932, 2020.
- [48] S.-Y. Wang, O. Wang, R. Zhang, A. Owens, and A. Efros. CNN-generated images are surprisingly easy to spot... for now. In *CVPR*, 2020.
- [49] Zhisheng Xiao, Karsten Kreis, and Arash Vahdat. Tackling the generative learning trilemma with denoising diffusion gans, 2022.
- [50] X. Xuan, B. Peng, W. Wang, and J. Dong. On the generalization of GAN image forensics. In *Chinese Conference on Biometric Recognition*, pages 134–141, 2019.
- [51] X. Zhang, S. Karaman, and S.-F. Chang. Detecting and Simulating Artifacts in GAN Fake Images. In *IEEE WIFS*, pages 1–6, 2019.
- [52] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning, 2020.