Deepfake detection using prior knowledge using angular frequency spectrum

Mariantonietta Maselli, Debora Pucciarelli

6 giugno 2025

Abstract

Negli ultimi anni, la diffusione di modelli generativi come GAN e Diffusion Models ha reso sempre più difficile distinguere le immagini reali da quelle sintetiche, sollevando rilevanti questioni di sicurezza, etica e affidabilità dell'informazione. In questo lavoro, si affronta il problema della classificazione automatica di immagini reali e generate artificialmente attraverso tecniche di apprendimento automatico. In particolare, si analizzano due approcci distinti: le Support Vector Machines (SVM), alimentate da feature estratte tramite Vision Transformer, e le reti neurali siamesi con triplet loss. A supporto dello studio è stata condotta un'analisi spettrale preliminare, basata sullo spettro angolare. I risultati sperimentali suggeriscono che entrambi gli approcci riescono a catturare pattern ricorrenti nelle immagini artificiali invisibili all'occhio umano, e che l'uso di modelli avanzati di classificazione può costituire una strategia efficace per il rilevamento automatico di contenuti generati.

Introduzione

Negli ultimi anni la generazione di contenuti sintetici ha registrato un'evoluzione impressionante, in particolare se si considera la facilità con cui possono essere create e modificate immagini. Queste tecnologie possono essere sfruttate positivamente in diversi settori come la sanità e la finanza, ma se si prendono in considerazione le problematiche etiche e sociali che derivano dall'uso improprio di tali strumenti (ad esempio a scopo di propaganda politica o disinformazione) diventa necessario ricercare dei modi per tutelarsi. [1]
I modelli generativi riescono ormai a raggiungere un livello di realismo tale da rendere indistinguibile un'immagine reale da un'immagine generata artificialmente, urge quindi creare e servirsi di ulteriori strumenti che possano compiere tale compito più efficacemente del solo occhio umano.
Alla luce di queste considerazioni, l'obiettivo di questo lavoro è stato quello di indagare la possibilità di distinguere tra immagini reali e immagini generate artificialmente mediante l'utilizzo di tecniche di apprendimento automatico. In particolare, ci si è concentrati su due approcci: le ***macchine a vettori di supporto*** (SVM) e le ***reti neurali siamesi*** (SNN). L'idea di fondo è stata valutare se questi modelli fossero in grado di apprendere pattern o caratteristiche ricorrenti nelle immagini sintetiche, impercettibili all'occhio umano, e sfruttarli per effettuare una classificazione accurata.

Stato dell'Arte

Rilevazione e attribuzione dei Deepfake

Nel contesto di ***Deepfake Detection*** emergono due obiettivi principali. Il primo è la **rilevazione**, che mira a valutare la probabilità che un'immagine sia generata artificialmente. Il secondo è l'**attribuzione**, che va oltre: tenta di identificare quale specifico modello generativo abbia prodotto l'immagine. L'attribuzione non solo rafforza i risultati della rilevazione, ma ne migliora anche la comprensibilità. [1]
Rilevazione e attribuzione sfruttano le tracce distintive lasciate dal processo di creazione di ogni immagine, sia essa reale o sintetica. I dispositivi fisici lasciano segni rilevabili e allo stesso modo, anche i modelli generativi basati su intelligenza artificiale lasciano impronte digitali (*fingerprint*) uniche. Queste derivano dalle specifiche architetture, dai passaggi di elaborazione interna (come filtraggio, campionamento, pooling) e perfino dal dataset utilizzato per l'addestramento.
Queste tracce non sono visibili a occhio nudo e richiedono analisi statistiche o trasformazioni nel dominio delle frequenze per essere individuate. Possono manifestarsi come picchi anomali nello spettro di potenza dei residui di rumore, rendendo possibile la rilevazione anche nei casi di immagini apparentemente perfette.

Importanza dello Spettro Angolare nei Deepfake

Uno studio recente [2] si propone di identificare tratti unici che permettano di distinguere le immagini reali da quelle generate artificialmente da modelli come le ***Generative Adversarial Networks*** (GAN) e i ***Diffusion Models*** (DM).
Per confrontare immagini reali e fake è utile effettuare un'analisi dello ***spettro angolare***, definendo un immagine come un insieme di tanti piccoli dettagli che possono essere interpretate come "frequenze". Lo spettro angolare mostra da quali direzioni spaziali provengono questi dettagli, cioè indica in quali angoli dell'immagine ci sono più particolari concentrati. I grafici riguardanti lo spettro angolare delle immagini generate artificialmente possono presentare picchi o pattern irregolari in certe direzioni che non si trovano nelle immagini naturali, rendendolo quindi uno strumento utile per il rilevamento di Deepfake.

SVM

Le ***Support Vector Machines*** sono modelli di apprendimento supervisionato introdotti per la prima volta nel contesto della classificazione binaria. [3]
Le SVM si basano sull'idea di trovare un iperpiano che separi al meglio le diverse classi all'interno dello spazio delle caratteristiche, cercando di massimizzare il margine, cioè la distanza tra i punti delle due classi più vicini all'iperpiano. Questo approccio aiuta a garantire una buona generalizzazione del modello, riducendo il rischio di overfitting.
Un aspetto molto potente delle SVM è la possibilità di usare funzioni kernel, che permettono di affrontare anche situazioni in cui i dati non sono separabili linearmente. In pratica, i dati vengono "proiettati" in uno spazio di dimensione superiore, dove spesso è più facile tracciare un confine netto tra le classi. Grazie a questa capacità, le SVM si rivelano particolarmente efficaci in scenari complessi, come ad esempio la rilevazione di immagini sintetiche, dove la distinzione tra le classi richiede una rappresentazione più articolata e non lineare dei dati.
Le SVM sono state ampiamente utilizzate nella letteratura per la rilevazione di contenuti generati artificialmente, dimostrandosi efficaci nel classificare immagini sintetiche a partire da pattern deboli ma sistematici, in [4] un classificatore SVM è stato addestrato su feature visive estratte da volti manipolati per rilevare artefatti geometrici introdotti durante il processo di generazione, ottenendo risultati promettenti nella distinzione tra immagini reali e deepfake.

Vision Transformer Il ***Vision Transformer*** (ViT) è un modello di *deep learning* che porta il meccanismo di *self-attention*, tipico del *natural language processing*, nel contesto dell'elaborazione delle immagini. In pratica, ViT suddivide un'immagine in piccole porzioni (*patch*), le tratta come una sequenza di *token* e, grazie al meccanismo di *multi-head attention*, riesce a cogliere relazioni globali tra le diverse parti dell'immagine.
Nel presente lavoro è stato impiegato il modello `ViT-B/16`, pre-addestrato sul dataset `ImageNet1K` (`ViT_B_16_Weights.IMAGENET1K_V1` [5]), utilizzato come *feature extractor*. Le rappresentazioni generate da ViT sono state successivamente fornite a un classificatore `SVM`, incaricato di distinguere tra immagini reali e immagini sintetiche.
Questa pipeline è supportata da diversi studi recenti. In [6], ViT-B/16 è stato utilizzato per estrarre rappresentazioni visive da immagini di piante in condizioni di stress idrico, successivamente classificate tramite SVM. I risultati hanno evidenziato un'accuratezza del 95% confermando la validità del modello anche in scenari complessi e sensibili. In un altro lavoro [7], lo stesso approccio è stato applicato al riconoscimento di piante medicinali, dove l'uso delle feature estratte da ViT e la classificazione con SVM ha superato le performance delle reti convoluzionali tradizionali, evidenziando inoltre l'interessante contributo delle mappe di attenzione nella spiegazione del processo decisionale.
Questi risultati supportano la scelta metodologica del presente studio, che sfrutta ViT-B/16 per la rappresentazione delle immagini e un classificatore SVM per la distinzione tra contenuti autentici e generati artificialmente.

Reti Neurali Siamesi

Una ***rete neurale siamese*** è una particolare tipologia di rete che permette di confrontare tra loro due elementi, ad esempio due immagini, e dire se sono simili o no.
Essa è costituita da due reti identiche e collegate, che prendono due input differenti e li analizzano separatamente, eseguendo gli stessi calcoli e utilizzando gli stessi pesi, con l'obiettivo di ricavare le informazioni più importanti di ciascun input che successivamente verranno confrontate. Se le informazioni ottenute risultano simili allora gli input sono considerati simili, altrimenti vengono classificati come differenti. [8]
Questo tipo di approccio funziona bene nella stragrande maggioranza di scenari, ma presenta dei problemi: ad esempio se consideriamo due immagini che rappresentano la stessa categoria, cioè immagini appartenenti alla stessa classe, la rete può avere difficoltà a distinguere le piccole differenze tra esse.
Per superare questa limitazione è stato introdotto un approccio chiamato ***triplet loss***, il quale, a differenza dell'approccio tradizionale finora utilizzato, prende in input **tre** immagini quali:

- ***Anchor:*** *immagine di riferimento*
- ***Positive:*** *immagine della stessa classe dell'anchor*
- ***Negative:*** *immagine di classe diversa*

L'obiettivo è insegnare alla rete che l'*anchor* e il *positive* sono simili, mentre l'*anchor* e il *negative* sono differenti.
Per far sì che l'allenamento risulti efficace, bisogna scegliere con attenzione le triplette da dare in input alla rete. Esistono varie tecniche per effettuare tale scelta, le quali variano in base al dataset, all'architettura di rete e alle risorse computazionalmente disponibili.
Un approccio semplice da implementare, che non richiede strategie complesse per la selezione di esempi negativi particolarmente facili o difficili da distinguere, è il ***Random Triplet Mining***. Questo metodo risulta particolarmente adatto quando il dataset a disposizione è sufficientemente vario e bilanciato, poiché consente di ottenere buoni risultati anche senza un'accurata selezione dei triplet.
Questo significa che le triplette vengono create scegliendo casualmente immagini positive e negative, senza preoccuparsi se l'immagine negativa sia troppo semplice o troppo simile a quella positiva. [9]
Il funzionamento della *triplet loss* si basa sull'utilizzo della ***distanza euclidea***, impiegata per misurare la differenza tra le rappresentazioni vettoriali (*embedding*) delle immagini. L'obiettivo è quello di ridurre al minimo la distanza tra l'*anchor* e il *positive*, ovvero due immagini appartenenti alla stessa classe, e al contempo aumentare la distanza tra l'*anchor* e il *negative*, cioè un'immagine di classe diversa, garantendo il rispetto di un margine minimo di separazione tra i due. [10]
Il modo in cui vengono scelte le triplette è fondamentale, in quanto compromette l'intero funzionamento della rete. È stato dimostrato che il triplet mining può influenzare il risultato persino più della funzione di perdita stessa [11], e al tal proposito sono stati proposti metodi intelligenti che scelgono in automatico, durante l'addestramento, le triplette più utili per migliorare l'apprendimento [12].
È stato proposto un sistema [13] in cui gli autori hanno usato un dataset con lettere e numeri scritti a mano e una rete neurale basata su VGG, ovvero una rete neurale utilizzata per analizzare le immagini individuando bordi, forme, oggetti. Hanno scoperto che, prima di usare la triplet loss, conviene allenare la rete con una normale classificazione supervisionata, usando la cross-entropy, una metodologia che permette alla rete per capire quanto sbaglia durante l'allenamento. Questo pre-addestramento aiuta la rete a partire da pesi migliori, rendendo più facile e veloce imparare buone rappresentazioni durante la fase con triplet loss. [9]
Recentemente, l'integrazione dei *Vision Transformer* in architetture siamesi ha mostrato risultati promettenti in diversi ambiti di classificazione e riconoscimento. Ad esempio, in [14], è stato proposto un modello chiamato *Siamese Transformer Network* (STN) che utilizza due rami paralleli basati su ViT per estrarre sia caratteristiche globali che locali da immagini, migliorando le prestazioni in scenari di few-shot learning.

Metodologia

Dataset

Il dataset utilizzato per lo sviluppo è *Artifact* [15]. Lo scopo dei creatori è stato quello di fornire un dataset completo e che fosse in grado di perfezionare la generalizzazione dei detector di Deepfake. Per rendere ciò possibile sono stati integrati diversi generatori, oltre che immagini reali, includendo una collezione di categorie vasta e il più variegata possibile. Le categorie maggiormente presenti nel dataset riguardano gli esseri umani e i volti oltre che animali, veicoli, luoghi e arte.

Studio preliminare

Abbiamo utilizzato il tool open-source proposto come esperimento nel paper *"**Intriguing properties of synthetic images: from generative adversarial networks to diffusion models**"* [2] per analizzare diverse caratteristiche nel dominio delle frequenze delle immagini sintetiche tra cui lo spettro di Fourier, le mappe di autocorrelazione, lo spettro angolare e lo spettro radiale. L'obiettivo è stato quello di comprendere se il tool fosse in grado di generalizzare su un dataset diverso da quello utilizzato dai creatori stessi.
Come discusso in *Studio degli spettri*, l'utilizzo di *Artifact* ha riportato risultati ambigui per quanto riguarda gli spettri di Fourier e la matrice di autocorrelazione. Avendo notato risultati promettenti relativamente allo spettro angolare si è deciso di concentrarsi approfonditamente su quello, apportando una miglioria al codice per scalare i grafici e renderli quindi facilmente confrontabili tra di loro, così da rendere ancora più evidenti le differenze dei vari modelli generativi presi in considerazione.

Analisi spettro angolare

Analizzando lo script `generate_spectra.py` del progetto `SyntheticImagesAnalysis` [2], si nota la presenza della funzione `get_spectra`, che consente di effettuare un'analisi delle immagini nel dominio delle frequenze. A differenza di approcci più comuni che si focalizzano su colori, forme o contorni, questo metodo si concentra su aspetti più nascosti

dell'immagine, come pattern sottili o strutture più ampie, valutando con quale intensità essi sono presenti. In altre parole, cerca di cogliere la "firma" spettrale di ogni immagine, fornendo così una rappresentazione alternativa ma molto informativa del contenuto visivo.
Per rendere possibile questo tipo di analisi, è stato utilizzato un operatore aritmetico noto come ***Trasformata di Fourier***. Questo strumento permette di passare dal dominio spaziale, in cui le immagini sono rappresentate come insiemi di pixel, al dominio delle frequenze, dove ogni punto corrisponde a una specifica frequenza spaziale. In questo nuovo spazio, è possibile osservare quanto ciascuna componente frequenziale contribuisce alla composizione dell'immagine, offrendo così una prospettiva diversa e particolarmente utile per individuare pattern e strutture ricorrenti.
Nel codice è presente la funzione `get_fft2`, che si occupa di elaborare ciascuna immagine trasformandola nel dominio delle frequenze. In particolare, l'immagine viene prima convertita in scala di grigi e normalizzata, quindi viene applicata la **Trasformata di Fourier bidimensionale** (`np.fft.fft2`) lungo gli assi spaziali, ovvero le dimensioni dell'immagine. Il risultato è uno spettro complesso che descrive la distribuzione delle frequenze spaziali presenti. Infine, viene calcolato lo **spettro di potenza**, che evidenzia l'intensità con cui ogni frequenza contribuisce alla struttura dell'immagine, fornendo così una sorta di "firma spettrale" dei dettagli visivi.
Il risultato finale di questa trasformazione è una nuova rappresentazione dell'immagine, in cui ogni punto indica quanta energia è presente in corrispondenza di una specifica direzione e frequenza spaziale. In altre parole, l'immagine viene scomposta nei suoi componenti frequenziali, permettendo di analizzare in dettaglio le strutture ricorrenti e i pattern nascosti che la compongono.
In seguito viene chiamata la funzione `get_spectrum`, che calcola lo spettro radiale a partire dallo spettro di potenza ottenuto tramite la trasformata di Fourier. Questo spettro misura quanta energia è distribuita nelle diverse frequenze spaziali, considerando solo la distanza dal centro dello spettro (cioè il modulo della frequenza), senza tener conto della direzione.
Viene inoltre utilizzata la funzione `get_spectrum_angular` per calcolare lo spettro angolare, ossia misurare come l'energia dello spettro è distribuita rispetto all'orientamento (cioè in funzione dell'angolo, indipendentemente dalla distanza dal centro), evidenziando eventuali direzioni privilegiate nella distribuzione spettrale.
Analizzando più nel dettaglio la funzione che calcola lo spettro *angolare*, essa prende in input lo spettro di Fourier bidimensionale e, per semplificare l'analisi, effettua inizialmente una media sui tre canali dell'immagine (RGB), ottenendo così una rappresentazione in scala di grigi. Questa immagine viene poi normalizzata, dividendo per il numero totale di pixel, in modo da rendere possibile un confronto coerente tra spettri di immagini diverse. Successivamente, si costruisce una griglia di coordinate spaziali in frequenza tramite la funzione `np.fft.fftfreq` per ciascun asse, ottenendo così un sistema cartesiano nel dominio delle frequenze. Ogni punto della griglia viene poi convertito da coordinate cartesiane a coordinate polari, ricavando per ciascun punto il corrispondente *raggio* (frequenza) e *angolo* (direzione). L'intervallo angolare viene suddiviso in 16 settori, mentre la regione centrale dello spettro (associata alle frequenze più basse) viene esclusa poiché contiene principalmente l'informazione legata al colore medio dell'immagine. Per ciascun settore angolare, viene infine calcolata l'energia media, producendo così una descrizione della distribuzione direzionale delle frequenze. Questo processo viene ripetuto per tutte le immagini analizzate.
Alla fine, vengono calcolate sia la media degli spettri radiali e angolari, che fornisce un profilo spettrale medio, e sia la varianza degli stessi spettri, utile per valutare la variabilità interna al dataset. Il grafico angolare è rappresentato in coordinate polari per evidenziare l'eventuale direzionalità nelle frequenze. Per garantire la comparabilità visiva tra gli spettri angolari associati ai diversi modelli analizzati, è stata adottata una scala radiale costante in tutti i grafici polari. In particolare, il limite massimo del raggio è stato fissato a `0.024`, assicurando che eventuali differenze nella distribuzione spettrale non venissero né enfatizzate né attenuate da variazioni automatiche di scala.
Inoltre, al fine di migliorare la leggibilità e ridurre il rumore visivo, le etichette associate ai tick radiali sono state nascoste, pur mantenendo visibile la griglia radiale. Questo approccio fornisce un riferimento strutturale coerente. I tick sono stati comunque definiti a intervalli regolari mediante il comando `ax.set_rticks([0.004, 0.008, 0.012, 0.016, 0.020, 0.024])`, permettendo di percepire in modo intuitivo l'intensità del contenuto spettrale, senza sovraccaricare la visualizzazione.
Lo spazio delle frequenze viene suddiviso in 16 settori angolari equidistanti, da 0 a π, e per ciascun settore si calcola la media del contenuto spettrale ad alta frequenza presente in una fascia angolare ristretta attorno all'angolo considerato.
Per focalizzarsi sulle componenti strutturali più significative, le frequenze troppo basse—tipicamente associate a informazioni generiche come il colore di

fondo o le grandi forme—vengono eliminate tramite un filtro passa-alto. Questo consente di concentrare l'analisi sulle caratteristiche più fini e strutturate dell'immagine.
Il risultato di questo processo è un grafico polare in cui, per ogni direzione angolare, è rappresentata l'intensità del segnale spettrale. Un grafico che mostra forti picchi in specifiche direzioni può indicare la presenza di strutture artificiali o artefatti, frequentemente riscontrabili nelle immagini sintetiche. Al contrario, uno spettro angolare più uniforme o coerente con quello osservabile nelle immagini reali suggerisce una distribuzione delle frequenze più naturale e credibile.
Questa analisi preliminare si è rivelata particolarmente utile, in quanto ha permesso di individuare quali modelli generano spettri angolari simili (o dissimili) tra loro. Tali osservazioni hanno guidato la selezione dei modelli da impiegare nelle successive fasi di addestramento, sia per il classificatore SVM sia per la rete neurale siamese.

Set di immagini utilizzate

Sono stati effettuati diversi test selezionando alcuni dei modelli presenti nel dataset *Artifact*, in particolare sono state analizzate:

- In una prima analisi **coppie** di modelli che presentavano spettri angolari sia simili che dissimili (mostrate nella Tabella 1).
- Successivamente **quattro** diversi modelli (mostrati nella Tabella 2), con spettri angolari sia simili che dissimili tra loro, in modo da spronare il modello e la rete ad apprendere e a generalizzare meglio su più classi.

I dati sono stati suddivisi in base a diversi set di immagini. Per ciascun gruppo è stata effettuata una suddivisione in training, validation e test set, utilizzando tre diverse proporzioni di test size (0.1, 0.2 e 0.3). Il numero totale di immagini per ciascun gruppo e la loro distribuzione nei tre sottoinsiemi varia in funzione della frazione scelta per il test set, come dettagliato nelle tabelle stesse.

SVM

Costruzione del dataset e pre-processing Le immagini sono state raccolte da directory contenenti i dati generati da diversi modelli, ognuno accompagnato da un file `metadata.csv` contenente path e validità delle immagini. Le classi sono state associate ai modelli confrontati: ad esempio, in un confronto binario, un modello rappresentava la classe 0 e l'altro la classe 1.

Una classe customizzata `ImageFolder`, costruita con *PyTorch* [16], ha gestito:

- il caricamento dei dati,
- l'assegnazione delle etichette binarie o multiclassi,
- il filtraggio delle immagini valide,
- il bilanciamento delle classi tramite `RandomUnderSampler` per garantire equità nella distribuzione tra le categorie.

Ciascuna immagine è stata ridimensionata a 224×224 pixel tramite **interpolazione bicubica**, successivamente centrata e normalizzata utilizzando le statistiche di *ImageNet* [17] (`media = (0.485, 0.456, 0.406)`, `deviazione standard = (0.229, 0.224, 0.225)`), al fine di garantirne la compatibilità con i modelli pre-addestrati utilizzati.

Estrazione delle feature con Vision Transformer L'estrazione delle feature (**768** in totale) è stata effettuata mediante il modello pre-addestrato *Vision Transformer* (`vit_b_16`), utilizzando i pesi `ViT_B_16_Weights.IMAGENET1K_V1` [5]. Ogni immagine è stata processata per ottenere una rappresentazione vettoriale semantica ad alta dimensionalità (*feature embedding*), che costituisce l'input per la classificazione. I vettori sono stati memorizzati per successivi utilizzi e confronti.

Addestramento della SVM senza riduzione dimensionale In una prima fase, è stata addestrata una SVM direttamente sulle feature estratte dal ViT, senza applicare alcuna riduzione dimensionale. In questi esperimenti:

- sono stati testati diversi valori di *test size* (`0.1, 0.2, 0.3`) per osservare la stabilità e generalizzazione del modello,
- è stato impiegato un **kernel RBF**,
- l'ottimizzazione degli iperparametri `C` e `gamma` è stata effettuata mediante `GridSearchCV`.

Addestramento della SVM con PCA Nella fase successiva, per mitigare l'elevata dimensionalità delle feature (e ridurre l'overfitting), è stata introdotta una riduzione dimensionale tramite ***Principal Component Analysis***. La PCA è stata applicata dopo la suddivisione train/test per evitare data leakage, ed è stata tarata per mantenere un numero sufficiente di componenti principali da spiegare la maggior parte della varianza (ad esempio 95%).
I dati trasformati sono stati poi utilizzati per addestrare nuovamente la SVM, seguendo lo stesso schema degli esperimenti precedenti. Anche in questo caso, sono stati effettuati esperimenti multipli variando il test size e rieseguendo il tuning degli iperparametri.

Tabella 1: *Distribuzione delle immagini per coppie di modelli*

Set di immagini	Spettro	Categoria	Totale	Test Size	Train	Validation	Test
`gansformer` `stylegan2`	Dissimili	Fake	20000	0.1	15999	2001	2000
				0.2	14000	2000	4000
				0.3	11999	2001	6000
`gau_gan` `generative_impainting`	Simili	Fake	14000	0.1	11200	1400	1400
				0.2	9800	1400	2800
				0.3	8399	1401	4200
`generative_impainting` `latent_diffusion`	Simili	Fake	20000	0.1	15999	2001	2000
				0.2	14000	2000	4000
				0.3	11999	2001	6000
`latent_diffusion` `stable_diffusion`	Dissimili	Fake	20000	0.1	15999	2001	2000
				0.2	14000	2000	4000
				0.3	11999	2001	6000

Tabella 2: *Distribuzione delle immagini per 4 modelli*

Set di immagini	Spettro	Categoria	Totale	Test Size	Train	Validation	Test

`generative_impainting`
`latent_diffusion`
`pro_gan`
`gau_gan`
	Simili	Fake	28000	0.1	22400	2800	2800

				0.2	19600	2800	5600
				0.3	16799	2801	8400

`stylegan2`
`taming_transformer`
`sfhq`
`imagenet`
	Dissimili	Fake (primi 3), True (imagenet)	40000	0.1	31999	4001	4000

				0.2	28000	4000	8000
				0.3	23999	4001	12000

Valutazione delle performance Per ogni configurazione (con e senza PCA, binaria e multiclasse, test size variabili), il modello SVM è stato addestrato e valutato separatamente. La valutazione è stata eseguita su dati mai visti durante l'addestramento, e le metriche sono state calcolate tramite `classification_report` e `confusion_matrix` di *scikit-learn*. [18]
Nel caso dell'esperimento multiclasse, si è adottata la strategia *one-vs-one* integrata nel classificatore `SVC`, che permette di estendere l'uso delle SVM a scenari con più di due classi. Anche in questo contesto, sono state valutate due diverse pipeline: una con riduzione dimensionale tramite *PCA* e una senza, al fine di confrontare l'impatto della compressione dello spazio delle feature sulle prestazioni del modello.

Rete neurale siamese

Dataset e pre-processing Le immagini di training e validation sono state preprocessate mediante **ridimensionamento a** 224×224 **pixel**, **normalizzazione** e **conversione in tensori**. In questo contesto, un *tensore* è una struttura dati multidimensionale (simile a una matrice, ma estendibile a più dimensioni) utilizzata per rappresentare numericamente le immagini in un formato compatibile con i modelli di *deep learning*. Il dataset è stato inoltre strutturato in modo da mantenere il bilanciamento tra le classi.

Estrazione delle feature Per l'estrazione delle caratteristiche visive, è stato utilizzato il modello ViT-B/16 preaddestrato su *ImageNet-21k* (`google/vit-base-patch16-224-in21k`) [19], che rappresenta un'implementazione dei Vision Transformer. Questo modello suddivide un'immagine in patch di dimensione 16×16 pixel, le trasforma in vettori tramite embedding lineare e le elabora come una sequenza, analogamente ai token nel NLP. Il meccanismo di self-attention consente di catturare relazioni globali tra le patch, permettendo al modello di apprendere rappresentazioni visive ricche e contestuali. Il token [CLS], aggiunto all'inizio della sequenza, aggrega le informazioni globali dell'immagine e viene utilizzato per compiti di classificazione.

Costruzione del Dataset Triplet Una volta ottenuti gli embedding, è stato costruito un dataset di tipo triplet, contenente terne di embedding (*anchor*, *positivo*, *negativo*). Le immagini "anchor" e "positive" appar-

tengono alla stessa classe, mentre quella "negative" a una classe differente. Le triplette sono state generate dinamicamente per coprire uniformemente tutte le classi.

Metodologia di scelta delle triplette La selezione delle triplette di embedding per l'addestramento con triplet loss è stata gestita in modo dinamico dalla classe `TripletDataset` nella quale è stato implementato il metodo `__getitem__`, che viene chiamato ogni volta che il modello ha bisogno di nuovi dati per l'addestramento, e fa si che il dataset generi una nuova tripletta in modo casuale.
Il processo si articola in tre fasi:

- A ogni chiamata a `__getitem__`, viene scelto un campione casuale che diventa l'*anchor* e vengono recuperati vari parametri come l'embedding di questo campione (`anchor_emb`), la sua etichetta (`anchor_label`) e il relativo indice nel dataset (`anchor_idx`);
- Per la scelta di un campione *positivo*, si considerano tutti i campioni che hanno la stessa etichetta dell'*anchor* i quali vengono salvati in una lista chiamata `self.groups`. Successivamente viene copiata la lista e si toglie l'*anchor* stesso così da non rischiare di scegliere lo stesso campione come positivo, poi si prende a caso uno degli altri e si recupera il suo embedding;
- Per quanto riguarda la scelta del *negativo*, si prendono tutti i campioni che non sono della stessa classe dell'*anchor* e si mettono insieme in una lista. Da qui si sceglie uno a caso da questa lista e si prende il suo embedding.

Il metodo `__getitem__` restituisce poi le triplette già pre-computate, cioè generate e salvate prima di iniziare l'addestramento, in modo da non doverle calcolare ogni volta durante il caricamento dei dati.

Addestramento della rete siamese La rete di proiezione (`EmbeddingNetwork`) è stata addestrata utilizzando la funzione di perdita ***Triplet Margin Loss***, con un margine impostato a `1.0`. Per l'ottimizzazione è stato impiegato l'algoritmo *Adam* con un learning rate pari a `1e-4`, per un totale di **15 epoche**. Durante l'addestramento sono stati salvati due set di pesi del modello: uno corrispondente al miglior valore di F1-score sulla validation e uno relativo alla validation loss più bassa.

Classificazione con similarità e K-NN ponderato Durante la validation e il test, ogni embedding di input viene confrontato con gli embedding del training set tramite la distanza euclidea. Le predizioni vengono calcolate adottando una strategia di *k-nearest neighbors* (`k = 5`) con pesi esponenzialmente decrescenti in base alla distanza ($\exp(-\alpha \cdot d^2)$ con $\alpha = 0.5$) e rinforzando i vicini più stretti ($\beta = 3$). La classe predetta è quella che ottiene il punteggio di somma pesata più alto tra i vicini.

Valutazione Il sistema è stato valutato su un insieme di test contenente immagini sintetiche generate dagli stessi quattro modelli. Le performance sono state misurate in termini di *accuracy*, *precision*, *recall* e *F1-score* con media macro. Inoltre, sono state prodotte rappresentazioni bidimensionali e tridimensionali degli embedding tramite t-SNE, PCA e UMAP per valutare visivamente la separabilità tra le classi nel nuovo spazio appreso dalla rete siamese.

Risultati

Studio degli spettri

Durante l'analisi iniziale, gli **spettri di Fourier** e le **matrici di autocorrelazione** si sono rivelati strumenti poco efficaci per distinguere visivamente tra i diversi modelli generativi. Sebbene in teoria questi strumenti offrano informazioni sulla distribuzione spaziale e frequenziale delle immagini, nella pratica si è osservato che le differenze tra i modelli erano estremamente sottili e poco marcate. Le variazioni presenti erano talmente minime che solo con uno sforzo attento e prolungato si riuscivano a cogliere, rendendo l'analisi soggettiva e di difficile interpretazione.
Al contrario, gli **spettri angolari** hanno mostrato fin da subito una maggiore capacità discriminativa: le differenze tra i modelli risultavano visibili anche a colpo d'occhio, con andamenti angolari distinti e spesso marcati. Questo ha motivato un cambiamento di strategia, portando a focalizzarsi esclusivamente sull'analisi angolare. A tal fine, è stata abbandonata anche la considerazione dello **spettro radiale**, per concentrare l'intero studio sulla rappresentazione spettrale che si è dimostrata più informativa e coerente con gli obiettivi dell'esperimento.
I risultati dell'esecuzione del Tool per la generazione degli spettri angolari dei modelli elencati nelle tabelle 1 e 2 (e ulteriori modelli non utilizzati nei successivi addestramenti) sono visibili nella Figura 1.
L'analisi visiva degli spettri angolari evidenzia una distinzione parziale tra immagini sintetiche e reali. Alcuni modelli deepfake, come *ProGAN* e *StyleGAN2* presentano spettri caratterizzati da simmetrie radiali e distribuzioni angolari regolari, spesso assenti nelle immagini reali. Tuttavia, non tutti i modelli generativi mostrano tracce così evidenti: ad esempio i risultati ottenuti con *Latent Diffusion* e *GauGAN* appaiono visivamente più simili a quelli dei dataset reali, suggerendo una maggiore efficacia nella riproduzione della complessità angolare naturale.

Un risultato interessante emerge dal confronto tra le immagini reali di *CelebA-HQ* e alcune immagini sintetiche: gli spettri angolari appaiono quasi sovrapponibili, indicando che in presenza di contenuti altamente strutturati o ripetitivi (ad esempio volti centrati su sfondo uniforme), anche le immagini reali possono produrre distribuzioni spettrali che ricordano quelle artificiali. Questo limita l'efficacia dell'analisi puramente visiva e sottolinea l'importanza di un approccio quantitativo o di feature statistiche per una classificazione robusta.

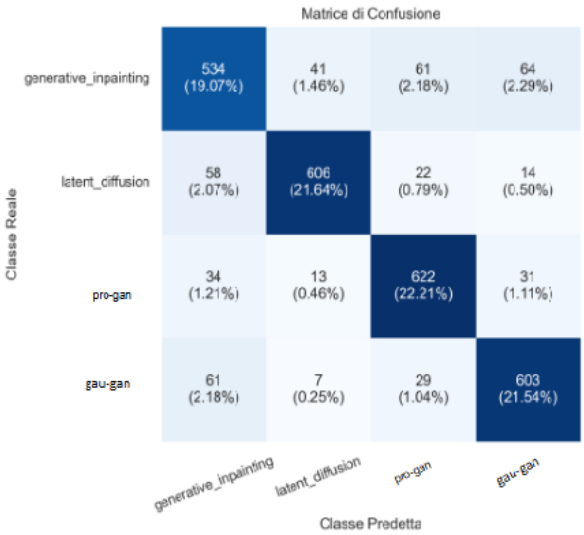

Figura 2: *Modelli con spettro angolare simile*

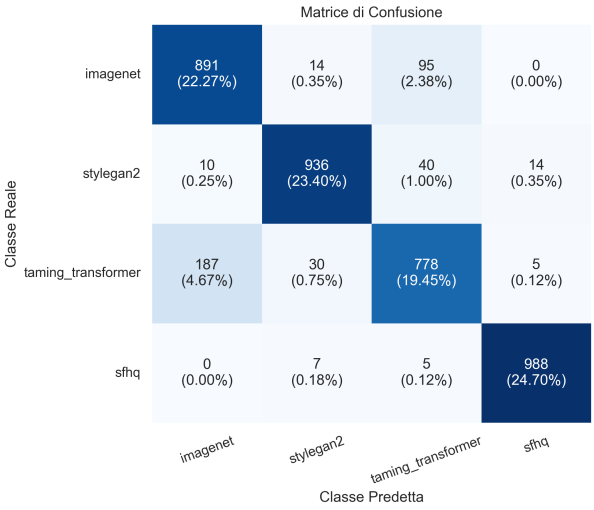

Figura 3: *Modelli con spettro angolare dissimile*

Addestramento con SVM

I risultati relativi all'addestramento con SVM sono riassunti nella Tabella 3. Sono state inoltre generate delle matrici di confusione per una visualizzazione più chiara delle prestazioni ottenute. In particolare, le Figure 2 e 3 mostrano le matrici di confusione relative ai due gruppi composti dai quattro modelli generativi con i risultati migliori (utilizzando una `test size` pari a `0.1`).

Le metriche di valutazione (*accuracy*, *precision*, *recall* e *F1-score*) sono state calcolate adottando la media `macro`, in modo da dare pari peso a ciascuna classe indipendentemente dal numero di campioni presenti. Anche nella rappresentazione della confusion matrix è stato riportato il valore di `macro avg` per ciascuna metrica, offrendo una panoramica sintetica ma bilanciata delle prestazioni complessive del classificatore.

PCA La PCA è stata eseguita mantenendo il 95% della varianza cumulativa, il che ha portato a una riduzione del numero di dimensioni mediamente del 30–40%. Dal punto di vista delle prestazioni, l'impatto è stato trascurabile: tutte le metriche di valutazione (*Accuracy*, *Precision*, *Recall*, *F1-score*) si sono ridotte di circa l'1% rispetto ai risultati ottenuti senza PCA, suggerendo che i dati originari contengano una certa ridondanza informativa. Di conseguenza, i risultati dettagliati non sono stati riportati in tabella.

Questa osservazione indica che l'utilizzo della PCA può essere considerato una valida strategia per ottimizzare l'efficienza computazionale del sistema senza compromettere significativamente le prestazioni. Tuttavia, per questa analisi si è scelto di presentare i risultati completi solo nella configurazione senza riduzione, in quanto rappresentativa del comportamento del modello in condizioni di massima informazione.

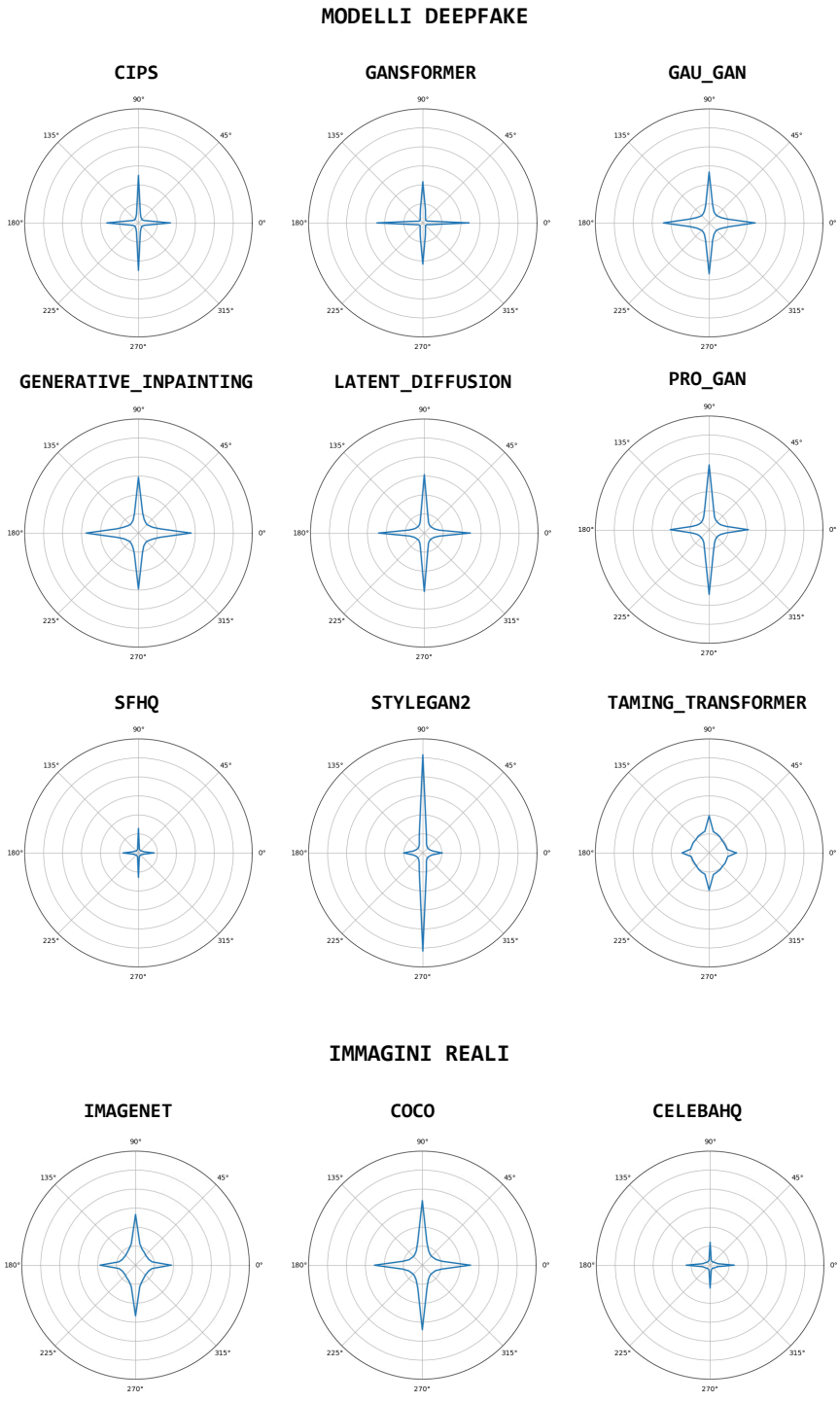

Figura 1: *Spettri angolari scalati per diversi modelli di Deepfake e immagini reali.*

Tabella 3: *Risultati di valutazione per coppie e gruppi di modelli con diversi test size*

Set di immagini	Spettro	Test Size	Accuracy	Precision	Recall	F1-score
gansformer stylegan2	Dissimili	0.1	0.98	0.98	0.98	0.98
		0.2	0.97	0.98	0.98	0.98
		0.3	0.97	0.98	0.98	0.98
gau_gan generative_impainting	Simili	0.1	0.89	0.90	0.90	0.90
		0.2	0.90	0.90	0.90	0.90
		0.3	0.89	0.90	0.90	0.90
generative_impainting latent_diffusion	Simili	0.1	0.91	0.92	0.92	0.92
		0.2	0.91	0.92	0.92	0.92
		0.3	0.91	0.92	0.92	0.92
latent_diffusion stable_diffusion	Dissimili	0.1	0.89	0.90	0.89	0.89
		0.2	0.89	0.90	0.90	0.90
		0.3	0.89	0.90	0.89	0.89
Gruppi di 4 modelli						
generative_impainting latent_diffusion pro_gan gau_gan	Simili	0.1	0.84	0.85	0.84	0.84
		0.2	0.84	0.84	0.84	0.84
		0.3	0.84	0.84	0.84	0.84
stylegan2 taming_transformer sfhq imagenet	Dissimili	0.1	0.89	0.90	0.90	0.90
		0.2	0.89	0.90	0.89	0.89
		0.3	0.89	0.89	0.89	0.89

Addestramento con rete neurale siamese

Modelli con spettri dissimili I risultati migliori ottenuti sui set di immagini generate da modelli con spettri dissimili sono riportati nella Tabella 4. Viene evidenziato il miglior modello in base a diverse metriche, tra cui *F1-score*, *accuracy*, *precision* e *Triplet Loss*.

Tra i modelli addestrati, quello corrispondente all'epoca 4 si distingue per aver ottenuto i valori più alti di *F1-score*, *accuracy* e *precision*, indicando una notevole capacità di distinguere efficacemente immagini appartenenti a coppie dissimili. Questo risultato suggerisce che il modello riesce non solo a identificare correttamente le differenze tra immagini reali e generate, ma anche a mantenere un buon equilibrio tra falsi positivi e falsi negativi, evidenziando un'ottima generalizzazione.

D'altro canto, il modello all'epoca 7 ha ottenuto la *validation loss* più bassa, segnalando una migliore coerenza nella separazione delle rappresentazioni nel tempo di validazione. Questo potrebbe indicare una strutturazione dello spazio degli embedding più solida dal punto di vista geometrico, anche se tale risultato non si riflette completamente nelle metriche di classificazione.

Pertanto, la selezione del modello migliore dipende dalla metrica ritenuta più significativa per l'obiettivo finale dell'applicazione: se si privilegia la qualità della classificazione, l'epoca 4 è preferibile; se invece si considera la struttura latente appresa, l'epoca 7 potrebbe risultare più interessante.

Tabella 4: *Prestazioni della rete siamese su modelli con spettri dissimili*

Ep.	Loss	Val.Loss	F1	Acc.	Prec.
4	0.0099	0.1631	**0.7101**	**0.7530**	**0.7575**
7	0.0033	**0.1466**	0.7040	0.7495	0.7485

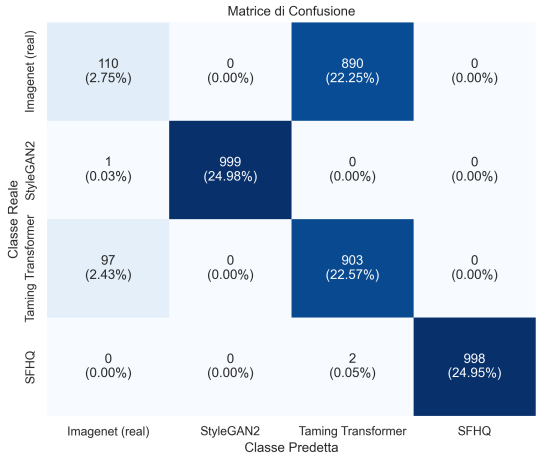

Figura 4: *Matrice di confusione rete siamese modelli con spettri dissimili*

La matrice di confusione visibile nella Figura 4 evidenzia la capacità della rete di distinguere correttamente i modelli. Si nota come si abbiano dei problemi nel riconoscimento per quanto riguarda il modello *Taming Transformer* a confronto con il set di immagini reali *Imagenet*, come evidenziato anche dalla visualizzazione mediante PCA nella Figura 7, che mostra un'evidente sovrapposizione tra i due set di immagini. Questo è stato attribuito al fatto che il modello è stato addestrato anche su *ImageNet*, come mostrato nel lavoro originale sul *Taming Transformer* [20].

Modelli con spettri simili Nella Tabella 5 sono mostrati i risultati relativi ai modelli addestrati su modelli con spettri simili. Le performance risultano inferiori rispetto al caso degli spettri dissimili, confermando la maggiore difficoltà del modello nell'apprendere rappresentazioni latenti efficaci per classi visivamente affini.

Nel dettaglio, il modello selezionato all'epoca 15 ha raggiunto un *F1-score* pari a circa 59,7%, indicando una difficoltà concreta nel discriminare correttamente tra immagini generate da modelli con caratteristiche spettrali simili. Tale risultato riflette l'ambiguità visiva e strutturale presente in questi modelli, dove le differenze tra le immagini risultano meno marcate e quindi meno informative per l'addestramento.

In aggiunta, la *validation loss* risulta sensibilmente più elevata rispetto al caso dei modelli dissimili, suggerendo una minore capacità della rete di strutturare lo spazio degli embedding in modo coerente. La maggiore sovrapposizione tra rappresentazioni latenti porta a una separazione meno netta tra classi, rendendo più complesso il processo di apprendimento discriminativo.

In sintesi, questi risultati sottolineano come la similarità spettrale tra modelli generativi impatti negativamente sulla capacità della rete siamese di apprendere distinzioni efficaci, portando a una generalizzazione più debole e a una maggiore ambiguità decisionale.

Tabella 5: *Prestazioni della rete siamese su modelli con spettri simili*

Ep.	Loss	Val.Loss	F1	Acc.	Prec.
15	0.0102	**0.5635**	**0.5973**	0.6320	0.6348

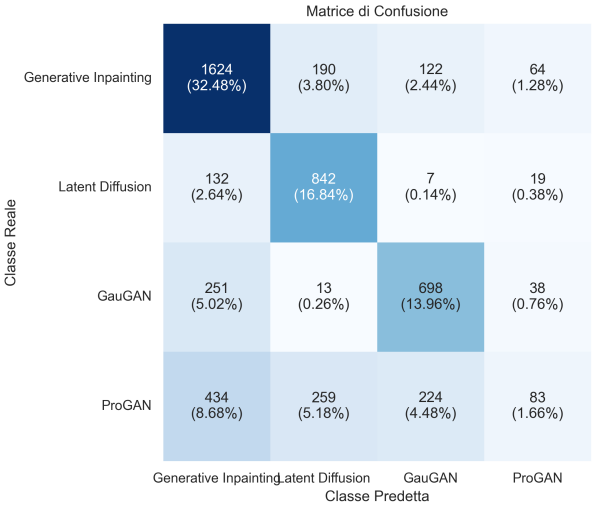

Figura 5: *Matrice di confusione rete siamese modelli con spettri simili*

La matrice di confusione riportata nella Figura 5 evidenzia una capacità predittiva limitata, specialmente nell'identificazione corretta delle classi *Generative Inpainting* e *Latent Diffusion* , dove si osserva un numero significativo di falsi positivi verso altre categorie.
In particolare, il modello dimostra una percentuale di accuracy globale insufficiente, con tassi di errore non trascurabili tra alcune coppie di classi. Queste confusioni possono essere attribuite alla forte somiglianza tra gli spettri angolari delle immagini generate rendendo difficile per il sistema discriminare con precisione tra le varie origini sintetiche.

Confronto Learning Curve

Il grafico della ***Learning Curve*** mostrato nella Figura 6 rappresenta l'andamento della training loss e della validation loss durante l'addestramento. Queste curve servono a monitorare come il modello sta imparando e, soprattutto, quanto riesce a generalizzare su dati mai visti.
In particolare, la **training loss** misura l'errore del modello sui dati usati per l'addestramento, mentre la **validation loss** valuta l'errore su un insieme di dati separato, non visto dal modello, utilizzato per simulare il comportamento su dati futuri o reali. Un buon modello dovrebbe mostrare una diminuzione

di entrambe le perdite, mantenendo la validation loss più alta della training loss ma comunque stabile o decrescente.

I modelli con spettri simili, come evidenziato dalle curve arancioni, presentano una tendenza significativa verso l'overfitting. Questo è visibile dallo stretto gap tra le curve di training loss e validation loss, che aumenta con il progresso delle epoche. La perdita di addestramento continua a diminuire fino a raggiungere valori molto bassi, mentre quella di validazione rimane costante intorno a `0.6`. Tale discrepanza suggerisce che il modello ha memorizzato troppo bene i dati di addestramento, senza generalizzare efficacemente sui dati di validazione. Questo fenomeno potrebbe essere attribuito al fatto che i dati fake utilizzati hanno uno spettro angolare simile, limitando la varietà del dataset e inducendo il modello a sovradattarsi ai pattern osservati nei dati di addestramento.

Al contrario, i modelli con spettri dissimili, rappresentati dalle curve blu, mostrano una dinamica diversa. Le curve di training loss e validation loss sono più vicine tra loro, indicando una migliore capacità di generalizzazione. In particolare, entrambe le curve tendono a convergere verso valori molto bassi, prossimi allo zero, dimostrando che il modello riesce a mantenere un buon livello di prestazioni sia durante l'addestramento che sulla validazione. Questo comportamento può essere spiegato dalla presenza di dati reali, che introducono una maggiore varietà nello spettro angolare rispetto ai dati fake. La combinazione di dati reali e fake fornisce al modello una rappresentazione più completa e diversificata dei dati, riducendo il rischio di overfitting.

Un ulteriore elemento da considerare è la quantità di dati disponibili per l'addestramento. Come ipotizzato, l'overfitting osservato nei modelli con spettri simili potrebbe essere dovuto alla scarsa quantità di dati utilizzati, poiché un numero insufficiente di campioni può indurre il modello a concentrarsi eccessivamente sui pattern specifici presenti nel dataset.

Il confronto tra i due insiemi di modelli evidenzia come la diversità nello spettro angolare dei dati possa influenzare significativamente le performance del modello. I modelli real con spettri dissimili dimostrano una migliore capacità di generalizzazione rispetto ai modelli con spettri simili, che sono più propensi all'overfitting.

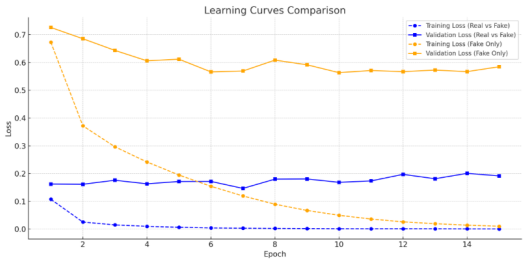

Figura 6: *Learning curve modelli con spettri simili e dissimili*

Visualizzazione 2D

Al fine di visualizzare le rappresentazioni apprese dal modello è stata implementata una funzione di visualizzazione bidimensionale degli *embedding*. Gli *embedding* generati nel test set sono stati proiettati in uno spazio 2D e 3D mediante tecniche di riduzione dimensionale (*PCA*, *t-SNE* e *UMAP*). Successivamente, i dati sono stati rappresentati graficamente utilizzando una mappa a dispersione, dove ciascun punto corrisponde a un campione e il colore indica la classe di appartenenza.

Modelli con Spettri Dissimili Un aspetto interessante emerso dall'analisi è la marcata sovrapposizione tra le immagini generate dal modello *Taming Transformer* e quelle appartenenti al dataset *ImageNet* visibile in Figura 7, composto da immagini reali. Questo comportamento, apparentemente controintuitivo considerando le differenze negli spettri angolari, può essere spiegato dal fatto che il modello è stato addestrato anche su *ImageNet*, generando quindi immagini che replicano fedelmente soggetti e strutture presenti nel dataset reale. Di conseguenza, la vicinanza nello spazio degli embedding riflette non solo la somiglianza semantica, ma anche la condivisione di caratteristiche visive apprese durante l'addestramento.

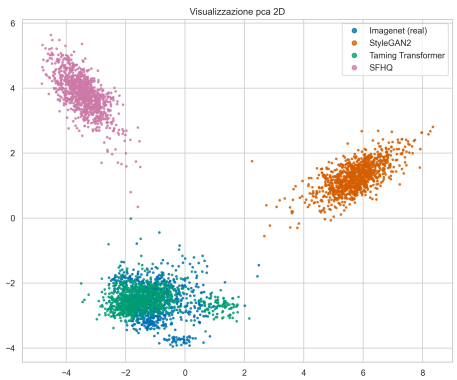

Figura 7: *Visualizzazione PCA 2D degli embeddings (modelli con spettri dissimili).*

Modelli con Spettri Simili La Figura 8 mostra la distribuzione bidimensionale degli embedding ottenuti tramite il modello siamese addestrato su immagini generate da modelli con spettro angolare simile. Come si può osservare, le classi risultano in gran parte sovrapposte, suggerendo che il modello fatica a discriminare efficacemente tra le diverse tipologie di immagini.
Questa difficoltà può essere attribuita sia alla somiglianza strutturale, già evidenziata dall'analisi spettrale, sia alla prossimità semantica tra i soggetti rappresentati. Quando le immagini, pur appartenendo a classi distinte, condividono caratteristiche visive e contenuti molto simili, il modello tende a collocarle in regioni contigue o sovrapposte dello spazio degli embedding, riducendo così la sua capacità di separazione.

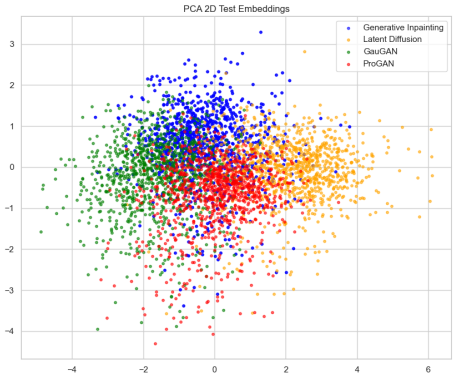

Figura 8: *Visualizzazione PCA 2D degli embeddings (modelli con spettri simili).*

Conclusioni

In questo lavoro è stato presentato un sistema composto da due prospettive metodologiche differenti ma reciprocamente integrabili per affrontare la stessa tipologia di problema, ovvero il riconoscimento e l'individuazione delle immagini Deepfake in relazione al modello generativo. Abbiamo utilizzato:

- ***Classificatore SVM:*** processo attraverso il quale una volta estratte le informazioni più rilevanti dall'immagine grazie a *Vit-16*, si utilizza un classificatore *SVM* per decidere a quale categoria appartiene;
- ***Rete neurale siamese addestrata con triplet loss:*** con questo approccio il sistema impara a mettere insieme le immagini simili mentre quelle diverse restano lontane. Quando viene aggiunta una nuova immagine, per classificarla si utilizza il metodo K-NN.

Entrambi gli approcci hanno mostrato una buona capacità di funzionamento, con prestazioni elevate soprattutto nel caso dei modelli con spettri tra loro dissimili, segno del fatto che lo spettro angolare può essere uno strumento utile. L'analisi spettrale ha avuto un'importanza fondamentale, permettendo di evidenziare pattern ricorrenti e direzionalità artificiali non percepibili a occhio nudo. Questo ha consentito di rafforzare la robustezza delle fasi successive di addestramento.
I risultati sperimentali svolti hanno mostrato come l'utilizzo dell'approccio *SVM* sia in grado di fornire ottime performance, specialmente quando gli si danno in pasto delle feature di alta qualità.
Al contrario, la rete siamese ha mostrato maggiori difficoltà quando i modelli erano molto simili in quanto le differenze spettrali e semantiche tra le immagini erano meno evidenti.
In generale, integrare l'analisi spettrale con con l'apprendimento automatico ha permesso di creare un sistema efficace per riconoscere e classificare i Deepfake.
Un ulteriore spunto di riflessione emerso dall'analisi dei risultati riguarda il potenziale utilizzo dello spettro come criterio guida nella costruzione stessa del dataset. L'analisi spettrale, infatti, può fornire indicazioni preliminari sulla varietà e la distribuzione delle caratteristiche frequenziali presenti nelle immagini. Questo può aiutare a selezionare esempi più informativi, a garantire una maggiore diversità spettrale all'interno del dataset, e a ridurre il rischio di bias legati a specifici generatori. In tal senso, lo spettro non rappresenta solo un mezzo per classificare, ma anche uno strumento per progettare in maniera più consapevole e robusta i dati di addestramento.

Sviluppi futuri

Un possibile sviluppo futuro potrebbe consistere in un'analisi più approfondita delle proprietà degli spettri angolari, al fine di valutare sistematicamente il loro contributo nel rilevamento dei Deepfake. Questo studio permetterebbe di comprendere se e in che misura tali rappresentazioni frequenziali possano costituire un indicatore discriminativo robusto e contribuire così alla definizione di strategie più efficaci e generalizzabili per la classificazione di contenuti sintetici.
Si potrebbe esplorare l'utilizzo di tecniche di triplet mining più avanzate per selezionare meglio i dati di allenamento, come l'adozione di un approccio adattivo che tenga conto delle informazioni spettrali. In questo scenario, la distanza spettrale potrebbe essere utilizzata come criterio guida per formare le triplette, migliorando la separabilità delle immagini

e riducendo il rischio di overfitting su determinate classi.
Inoltre, sarebbe interessante esplorare il miglioramento del modello *ViT* tramite l'impiego di immagini sintetiche, rafforzando così ulteriormente la capacità di generalizzazione del classificatore. L'uso di immagini generate potrebbe aumentare la diversità del dataset e consentire una rappresentazione più robusta delle immagini Deepfake, specialmente in presenza di varianti generatori non viste durante l'allenamento.
Infine, un altro ambito di ricerca potrebbe riguardare l'integrazione di modelli multimodali che combinano l'analisi visiva con altre informazioni, come i metadati o le caratteristiche audio (nel caso di video), per migliorare l'affidabilità e la precisione del sistema di rilevamento delle immagini Deepfake. Questo approccio potrebbe essere particolarmente utile per affrontare i Deepfake video, dove la sfida è ancora più complessa rispetto alle singole immagini statiche.

Riferimenti

[1] Diangarti Tariang et al. "Synthetic image verification in the era of generative artificial intelligence: What works and what isn't there yet". In: *IEEE Security & Privacy* (2024).

[2] Riccardo Corvi et al. "Intriguing properties of synthetic images: from generative adversarial networks to diffusion models". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2023, pp. 973–982.

[3] Corinna Cortes e Vladimir Vapnik. "Support-vector networks". In: *Machine learning* 20.3 (1995), pp. 273–297.

[4] Yuezun Li, Ming-Ching Chang e Siwei Lyu. "Exposing DeepFake videos by detecting face warping artifacts". In: *arXiv preprint arXiv:1811.00656* (2018).

[5] PyTorch Core Team. *torchvision.models.vit_b_16*. `https://docs.pytorch.org/vision/stable/models/generated/torchvision.models.vit_b_16.html`. Accessed: 2025-05-23. 2024.

[6] Jinxin You et al. "DroughtViT: vision transformer for early and accurate detection of drought stress in tomato plants". In: *Computers and Electronics in Agriculture* 213 (2024), p. 108342.

[7] V. N. Manjunath Aradhya, Anilkumar Patil e Harish Bhat. "Medicinal Plant Species Identification Using Vision Transformer and Explainable AI". In: *Proceedings of International Conference on Recent Trends in Machine Learning, IoT, Smart Cities and Applications*. Springer, 2024, pp. 47–60.

[8] Ruslan Salakhutdinov Gregory Koch Richard Zemel. "Siamese Neural Networks for One-shot Image Recognition". In: ().

[9] Yifan Wang et al. "Combining Negative Selection Techniques for Triplet Mining in Deep Metric Learning". In: (2022), pp. 1234–1243.

[10] Yifan Wang et al. "Combining Negative Selection Techniques for Triplet Mining in Deep Metric Learning". In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2022, pp. 1234–1243.

[11] Chao-Yuan Wu et al. "Sampling matters in deep embedding learning". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2840–2848.

[12] Ben Harwood et al. "Smart mining for deep metric learning". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2821–2829.

[13] Florian Schroff, Dmitry Kalenichenko e James Philbin. "Facenet: A unified embedding for face recognition and clustering". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 815–823.

[14] Weihao Jiang, Shuoxi Zhang e Kun He. "Siamese Transformer Networks for Few-shot Image Classification". In: *arXiv preprint arXiv:2408.01427* (2024).

[15] Md Awsafur Rahman et al. "Artifact: A large-scale dataset with artificial and factual images for generalizable and robust synthetic image detection". In: *2023 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2023, pp. 2200–2204.

[16] Adam Paszke et al. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. 2019. URL: `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

[17] Jia Deng et al. "ImageNet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. IEEE. 2009, pp. 248–255.

[18] Fabian Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[19] Google Research. *google/vit-base-patch16-224-in21k*. `https://huggingface.co/google/vit-base-patch16-224-in21k`. Accessed: 2025-05-27. 2022.

[20] Patrick Esser, Robin Rombach e Björn Ommer. “Taming transformers for high-resolution image synthesis”. In: *Proceedings of the IEEE/-CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 12873–12883.