



UNIVERSITÀ
DEGLI STUDI
FIRENZE

Dipartimento di Ingegneria dell'Informazione

Tesi di Laurea triennale in Ingegneria Informatica

Anomaly detection con un autoencoder variazionale: una strategia per riconoscere i deepfakes

Anomaly detection with a variational autoencoder: a strategy to expose deepfakes

N° matricola: 7075054

Relatore: Andrew D. Bagdanov

Laureando: Niccolò Malgeri

Anno accademico: 2023-2024

Indice

1	Introduzione	2
2	Autoencoder variazionale: un potente modello per l'anomaly detection	3
2.1	Autoencoder: definizione e struttura	3
2.2	Autoencoder variazionali	3
3	Lavori correlati: OC-FakeDect	4
3.1	FaceForensics++	5
3.2	OC-FakeDect: funzione di perdita	5
3.3	Il trucco di riparametrizzazione	6
3.4	OC-FakeDect1 e OC-FakeDect2	6
4	Il nostro approccio	7
4.1	Scelta del dataset e struttura del modello	7
4.1.1	ResNet: definizione e struttura	7
4.1.2	Metriche e scelte implementative	8
5	Descrizione degli esperimenti condotti	8
5.1	Preparazione dei dati e fase di addestramento	8
5.2	Test effettuati	8
6	Presentazione ed analisi dei risultati sperimentali	10
6.1	Iistogramma degli score di ricostruzione	10
6.2	Curva precisione-richiamo	11
6.3	Rappresentazione t-SNE dello spazio latente	11
6.4	Errori di ricostruzione	12
6.4.1	ResNetOCVAE1	13
6.4.2	ResNetOCVAE2	15
7	Confronto con OC-FakeDect1	15
8	Conclusioni e lavori futuri	17
9	Ringraziamenti	18
10	Bibliografia	19

1 Introduzione

Al giorno d'oggi è evidente come il progresso ed il continuo avanzamento della tecnologia abbiano portato numerosi benefici alla società moderna, così com'è altrettanto evidente quali possono essere i rischi di un utilizzo tecnologico scorretto. Un esempio concreto e molto attuale è sicuramente rappresentato dai **deepfake**, ossia immagini, audio o video sintetici creati utilizzando tecniche di intelligenza artificiale in modo tale da sembrare reali, solitamente allo scopo di recare danno ad un determinato soggetto oppure diffondere disinformazione.

Per comprendere quanto questi contenuti possono rappresentare un problema, basta pensare al recente caso della diffusione di materiale pornografico, generato artificialmente, raffigurante la nota cantante Taylor Swift [1], oppure alla registrazione vocale in cui il presidente americano Joe Biden incentiva i propri sostenitori a non prendere parte alle imminenti elezioni e di rinunciare al voto, anch'essa rivelatasi essere falsa e generata artificialmente [2].

Nel contesto del Deep Learning, tra i modelli all'avanguardia per la creazione di deepfake ci sono le GAN (Generative Adversarial Network) e gli autoencoder variazionali o **VAE (Variational AutoEncoder)**; in particolare, DCGAN (Deep Convolutional GAN) oppure WGAN (Wasserstein GAN) hanno mostrato notevoli risultati nella generazione di immagini e video sintetici [3], rendendo perciò necessario lo sviluppo di metodi per riconoscere correttamente il materiale multimediale generato da tali modelli; per esempio, C.C. Hsu, Y.X. Zhuang e Y.Y. Lee hanno proposto un valido approccio per rilevare immagini create utilizzando proprio DCGAN, WGAN e altri modelli generativi [4].

Questi rilevatori di deepfake (*deepfake detectors*) però mettono in evidenza una loro significativa limitazione, legata al task più comune per il quale essi vengono addestrati, ossia la classificazione binaria; infatti, per poter distinguere correttamente immagini/video reali da quelli sintetici, non è solamente necessario avere una grande quantità di dati, ma è anche necessario aggiungerne di nuovi e ripetere la fase di addestramento ogni qualvolta nascono nuove tecniche per la creazione di deepfake; vista la velocità con cui ciò avviene e dati gli eccessivi costi (in termini di tempo di esecuzione) necessari per addestrare nuovamente una rete neurale, soprattutto se complessa, l'approccio binario risulta essere poco efficace.

Una possibile alternativa è rappresentata dall'*anomaly detection* mediante classificazione a una classe (*one-class classification*), che consiste nell'addestrare un modello per prevedere correttamente, all'interno di un dataset, la classe di dati "normali", riconoscendo qualsiasi altro esempio come un'anomalia. Secondo un possibile approccio, si utilizzano dei deepfake detectors ai quali vengono fornite solo immagini reali durante la fase di addestramento, mentre per la fase di test vengono scelte sia immagini reali che deepfake. In questo modo, è possibile ottenere una precisione di classificazione elevata per quanto riguarda i dati normali (le immagini reali) e decisamente minore per le anomalie (i deepfake). Questo risultato permette al modello utilizzato di riconoscere con successo i dati appartenenti a ciascuna classe; inoltre, l'utilizzo di queste tecniche consente di evitare di ripetere l'addestramento in caso nascano nuovi metodi per la generazione di immagini sintetiche, poiché il modello riconosce come anomalia tutto ciò che rappresenta delle differenze significative rispetto alle immagini reali, indipendentemente dal processo di alterazione e manipolazione effettuato.

In questa tesi si cerca di realizzare la task dell'anomaly detection utilizzando come deepfake detectors due autoencoder variazionali, uno simmetrico e l'altro asimmetrico, basati entrambi sulla rete neurale ResNet-18. Le prestazioni degli autoencoder vengono valutate attraverso diverse analisi e numerosi test, con lo scopo di capire quanto può influire la complessità di modelli come questi sulla capacità di riconoscere con successo immagini generate artificialmente.

Vedremo che il modello asimmetrico non ottiene dei buoni risultati; la versione simmetrica, invece, riesce a superare, in termini di precisione, accuratezza e richiamo, le performance di uno dei modelli rappresentanti lo stato dell'arte per quanto riguarda la deepfake detection.

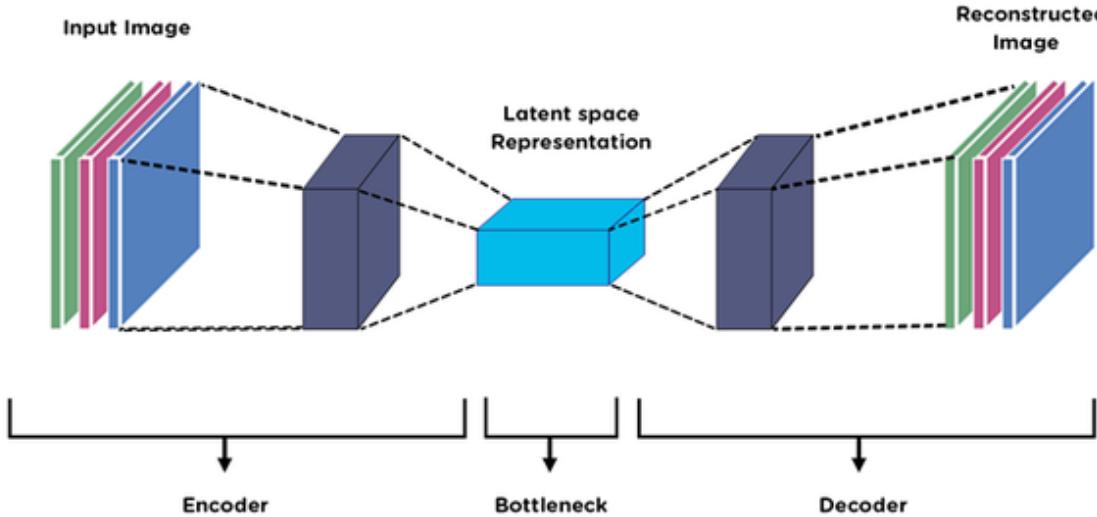


Figura 1: struttura di un autoencoder

2 Autoencoder variazionale: un potente modello per l'anomaly detection

Questa sezione contiene una descrizione delle caratteristiche generali degli autoencoder, seguita da una più precisa discussione sugli autoencoder variazionali, oggetti centrali dei nostri esperimenti.

2.1 Autoencoder: definizione e struttura

Gli **autoencoder** rappresentano una classe di modelli che permettono di comprimere dei dati in input per poi, successivamente, decomprimerli e ricostruirli nella loro forma originale. Come è possibile vedere in Figura 1, un autoencoder è costituito da tre componenti:

- **encoder**: è una rete neurale artificiale (di solito una rete neurale convoluzionale) che prende un dato in input, come per esempio un'immagine, e lo trasforma in un vettore riducendone la dimensionalità
- **bottleneck**: è lo spazio dei vettori rappresentanti la forma compressa dei dati iniziali. È anche denominato **spazio latente**
- **decoder**: è una rete neurale artificiale (di solito una rete neurale convoluzionale) che prende un vettore dello spazio latente e, tramite un processo chiamato *upsampling*, effettua la ricostruzione del dato originale corrispondente

2.2 Autoencoder variazionali

Nonostante gli autoencoder rappresentino dei validi sistemi per la compressione e decompressione dei dati, la loro capacità di ricostruzione è limitata ai campioni forniti al modello in ingresso; ciò significa che non è possibile generare nuovi dati a partire dalla sola decodifica di un vettore dello spazio latente, poiché non è noto al modello il criterio con cui selezionare tale vettore. È possibile superare questa difficoltà utilizzando una versione più "raffinata" dell'autoencoder, ossia l'**autoencoder variazionale** o **VAE** (*Variational AutoEncoder*).

A differenza della versione classica, in questo tipo di autoencoder ogni input x non viene mappato ad un singolo vettore z dello spazio latente, bensì ad una distribuzione. In particolare:

- l'encoder non calcola la funzione $f : x \rightarrow z$, ma $f : x \rightarrow q_\phi(z|x)$, dove ϕ identifica i parametri dell'encoder

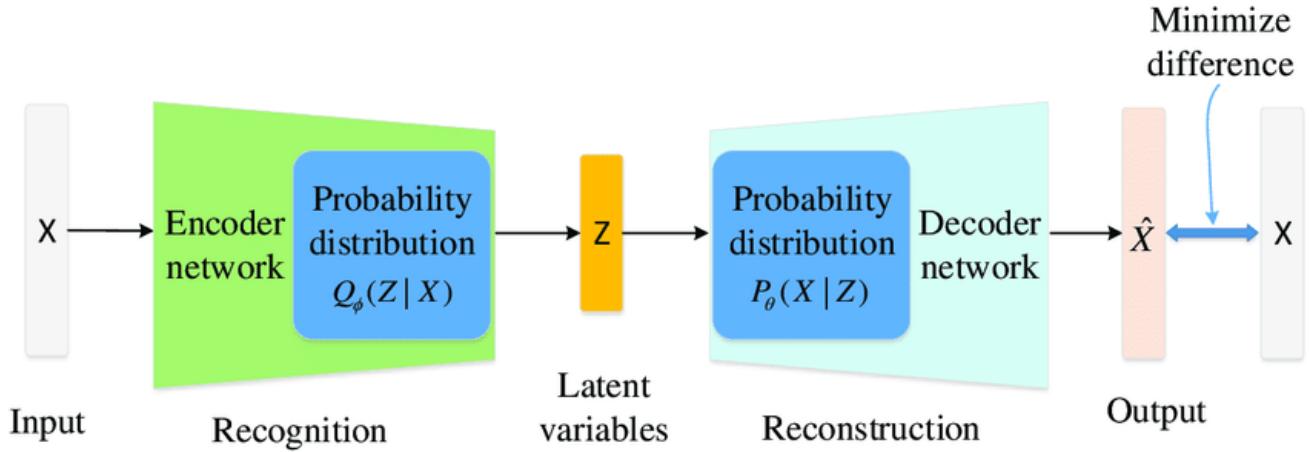


Figura 2: struttura di un autoencoder variazionale

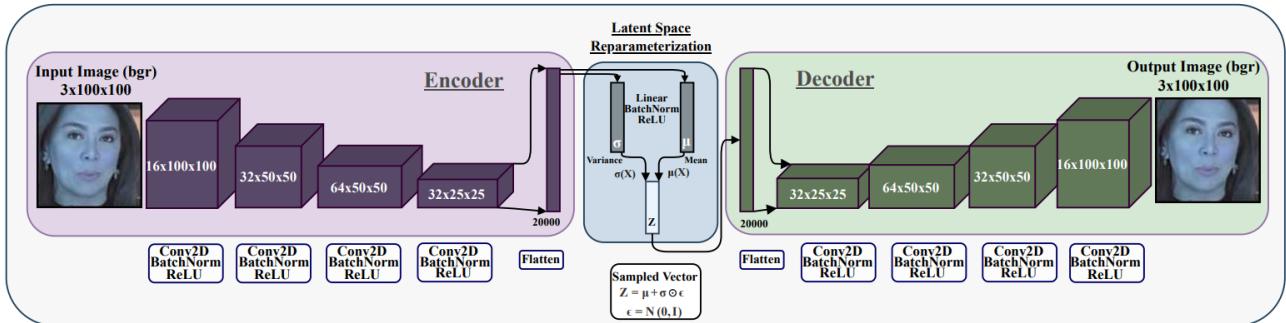


Figura 3: struttura di OC-VAE

- il decoder non calcola la funzione $g : z \rightarrow \hat{x}$, dove \hat{x} è l'output del modello, ma $g : z \rightarrow p_\theta(x|z)$, dove θ identifica i parametri del decoder

Encoder e decoder perciò hanno come obiettivo quello di approssimare, rispettivamente, una distribuzione a posteriori ed una (log) verosimiglianza.

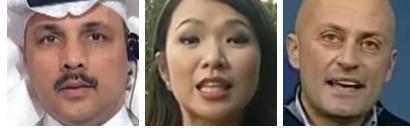
Come si può comprendere dall'osservazione della Figura 2, un buon autoencoder variazionale deve generare degli output che siano il più possibile simili ai corrispondenti input, cercando anche di rispettare le condizioni sulle distribuzioni che encoder e decoder devono approssimare. Perciò, con questi modelli si pone come obiettivo la minimizzazione della seguente funzione di perdita (*loss function*):

$$L(\phi, \theta, x) = KL(q_\phi(z|x), p_\theta(z)) + \mathbb{E}_{q_\phi(z|x)}[p_\theta(x|z)] \quad (1)$$

dove il primo termine è detto **divergenza di Kullback-Leibler** e misura la differenza tra la distribuzione a posteriori e quella a priori ($p_\theta(z)$) dei vettori z dello spazio latente, mentre il secondo termine misura l'**errore di ricostruzione** del VAE, ossia la differenza tra l'input dell'encoder x e l'output ricostruito del decoder \hat{x} .

3 Lavori correlati: OC-FakeDect

Un importante lavoro da menzionare utilizzato come punto di partenza per ciò che viene discusso in questa tesi è sicuramente "OC-FakeDect: Classifying Deepfakes Using One-Class Variational Autoencoder" di H. Kalid e S.S. Woo [5]; questo paper di ricerca, come suggerisce il titolo, presenta OC-FakeDect, un autoencoder variazionale per il riconoscimento di immagini deepfake, addestrato per un task di anomaly detection con classificazione a una classe. In Figura 3 viene mostrato il VAE utilizzato dagli autori come riferimento per lo sviluppo di OC-FakeDect, denominato OC-VAE. Nei prossimi paragrafi viene brevemente descritto il dataset su cui OC-FakeDect è stato addestrato; la discussione dei lavori correlati viene conclusa dalla descrizione delle metriche del modello utilizzate, le quali sono fondamentali per comprendere l'approccio al problema da noi seguito.



(a) Immagini reali



(b) Immagini artificiali estratte dalla forgery *Deepfakes*

Figura 4: Esempi di immagini in *FaceForensics++*

3.1 FaceForensics++

Come dataset per gli esperimenti riguardanti OC-FakeDect è stato scelto *FaceForensics++* [6]; esso contiene più di 1000 video reali raffiguranti volti umani, reperiti principalmente da YouTube, uniti ad un elevato numero di immagini deepfake ottenute applicando diverse tecniche di manipolazione alle sequenze originali e successivamente estraendo da queste ultime i frame che le compongono. Le immagini artificiali presenti nel dataset sono suddivise in quattro gruppi, o "forgery", distinti in base alla tipologia di tecnica utilizzata:

- **FaceSwap**: contiene immagini in cui il volto del soggetto è stato sostituito con quello presente in una seconda immagine (*face replacement*). Il procedimento di alterazione viene effettuato attraverso programmi di computer grafica che permettono di estrarre i landmarks del volto dell'immagine sorgente ed applicarli all'immagine target;
- **Face2Face**: contiene immagini ottenute con modalità simili a *Face2Face*, con la differenza che i tratti originali del volto nell'immagine target sono conservati (*face reenactment*)
- **Deepfakes**: questa forgery implementa le tecniche utilizzate in *FaceSwap*, ma le realizza mediante un approccio "learning-based", ossia sfruttando reti neurali profonde per automatizzare il processo di creazione dei deepfake;
- **NeuralTextures**: similmente a *Deepfakes*, le immagini di *NeuralTextures* vengono create attraverso un approccio learning-based, utilizzando però le tecniche di manipolazione e di rilevamento dei tratti dei volti di *Face2Face*

In un secondo momento, è stata aggiunta una quinta forgery denominata *Deepfake-detection*, contenente circa 3000 video artificiali raffiguranti 28 attori pagati da Google per registrare le sequenze originali. Alcuni esempi di immagini contenute in FaceForensics++ si possono osservare in Figura 4.

3.2 OC-FakeDect: funzione di perdita

In OC-FakeDect, il campionamento di un vettore z dallo spazio latente viene effettuato mediante altri due vettori μ e σ , i quali rappresentano media e deviazione standard della distribuzione a posteriori, che in questo caso è una distribuzione normale:

$$z \sim \mathcal{N}(\mu, \sigma^2) \quad (2)$$

Di conseguenza, la funzione di perdita assume la seguente forma:

$$\begin{aligned} L(\theta, \mu, \sigma, x) &= KL(\mathcal{N}(\mu(x), \sigma^2(x)), \mathcal{N}(0, 1)) + \|x - p_\theta(x|z)\|^2 \\ &= -\frac{1}{2}(1 + \log(\sigma^2(x)) - \mu^2(x) - \sigma^2(x)) + \|x - p_\theta(x|z)\|^2 \end{aligned} \quad (3)$$

E' importante osservare che, in questo caso, l'errore di ricostruzione viene calcolato come somma residua dei quadrati (*RSS*, *Residual Sum of Squares*) tra l'input x e l'output ricostruito del decoder.

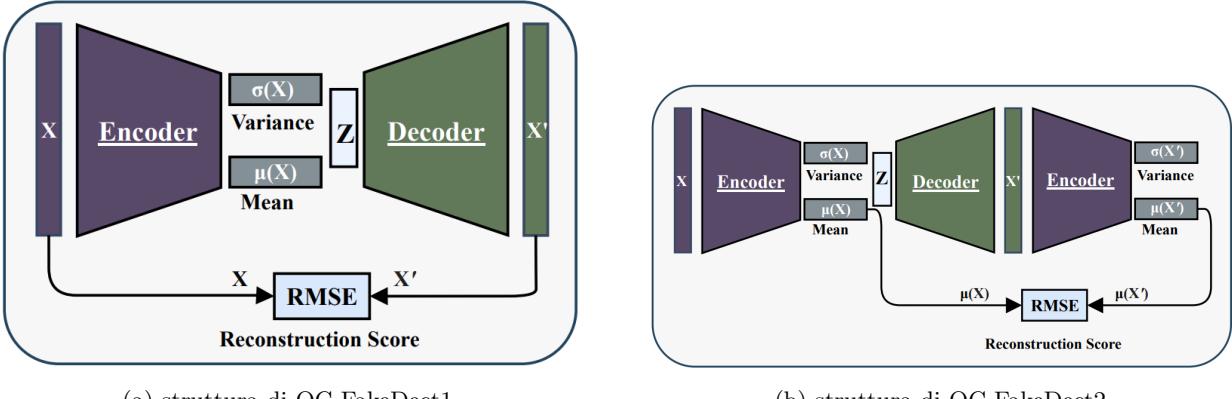


Figura 5: Due differenti versioni del VAE OC-FakeDect

3.3 Il trucco di riparametrizzazione

La funzione di perdita, definita nel precedente paragrafo, evidenzia il fatto che la scelta di z rende impossibile l'aggiornamento dei pesi del modello attraverso la *backpropagation*, poiché quest'ultima si basa sul calcolo deterministico di gradienti e non può essere applicata se sono presenti variabili aleatorie. Per risolvere tale inconveniente, viene utilizzato il **trucco di riparametrizzazione** (*reparametrization trick*):

$$z = \mu(z) + \sigma(z) * \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, 1) \quad (4)$$

Questa tecnica consiste nel separare la componente stocastica del campionamento dallo spazio latente da quella deterministica, attraverso l'introduzione della variabile aleatoria ε ; in questo modo, infatti, il calcolo di z diventa un'operazione differenziabile, consentendo l'addestramento del modello in modo simile a come si farebbe per un autoencoder tradizionale.

3.4 OC-FakeDect1 e OC-FakeDect2

All'interno del paper di ricerca, gli autori propongono due versioni di autoencoder variazionale, ossia OC-FakeDect1 e OC-FakeDect2, le cui caratteristiche sono di seguito brevemente discusse:

- **OC-FakeDect1:** rappresenta la versione di VAE descritta finora nella Sezione 3 e denominata OC-FakeDect. La metrica principale per valutare le prestazioni del modello (in Figura 5a) è lo **score di ricostruzione** (*reconstruction score*), calcolato come radice dell'errore quadratico medio (RMSE, Root Mean Squared Error) tra ogni immagine di input x e la corrispondente immagine di output \hat{x} :

$$\text{reconstruction_score}(x, \hat{x}) = \sqrt{\frac{1}{CN} \sum_{c=1}^C \sum_{i=1}^N (x_{ci} - \hat{x}_{ci})^2} \quad (5)$$

dove i e c fanno riferimento rispettivamente ad un determinato pixel e ad un determinato canale di x e \hat{x} , mentre N indica il numero di pixel per canale

- **OC-FakeDect2:** a differenza della prima versione, OC-FakeDect2 (in Figura 5b) contiene un secondo encoder identico al primo, che prende in ingresso l'output \hat{x} ricostruito del decoder e genera due ulteriori vettori $\mu(\hat{x})$ e $\sigma(\hat{x})$. Lo score di ricostruzione viene quindi calcolato come radice dell'errore quadratico medio tra la rappresentazione latente di ogni immagine di input e quella relativa alla corrispondente immagine di output:

$$\text{reconstruction_score}(x, \hat{x}) = \sqrt{\frac{1}{CN} \sum_{c=1}^C \sum_{i=1}^N (x_{\mu ci} - \hat{x}_{\mu ci})^2} \quad (6)$$

dove $x_{\mu ci}$ e $\hat{x}_{\mu ci}$ sono l'equivalente di x_{ci} e \hat{x}_{ci} , ma relativi a $\mu(x)$ e $\mu(\hat{x})$ rispettivamente

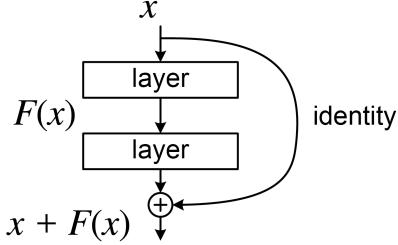


Figura 6: esempio di blocco residuo

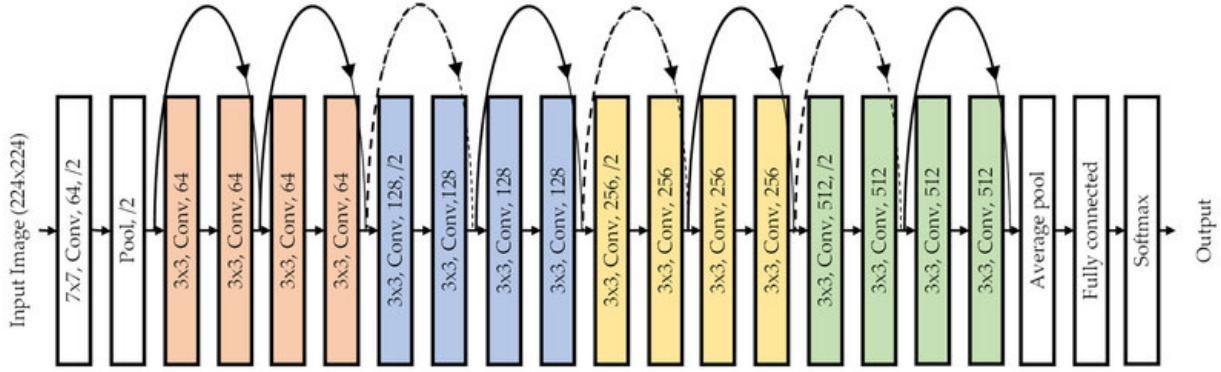


Figura 7: struttura di ResNet-18

4 Il nostro approccio

In questa sezione viene descritto il nostro approccio al problema del riconoscimento dei deepfakes, a partire dalla scelta del dataset ed alla discussione sulla struttura del modello utilizzato, fino alla spiegazione dettagliata degli esperimenti e test condotti tramite esso.

4.1 Scelta del dataset e struttura del modello

Per cercare di allinearsi con gli autori di OC-FakeDect, abbiamo scelto come dataset per i nostri esperimenti *FaceForensics++*, limitandoci però ad utilizzare la forgery *Deepfakes*; infatti, non potendo determinare a priori l'esito degli esperimenti, abbiamo considerato opportuno iniziare da un dataset relativamente ridotto, per poi eventualmente estenderlo ad altre forgery nel caso in cui i risultati ottenuti fossero stati positivi.

Per quanto riguarda l'autoencoder variazionale, abbiamo preso come riferimento OC-FakeDect1, modificando tuttavia la struttura dell'encoder e del decoder in modo che il primo simulasse la rete neurale *ResNet-18*, una versione di ResNet composta da 18 layer convoluzionali (Figura 7), mentre il secondo ripetesse le operazioni di codifica in ordine inverso per ricostruire l'immagine in input. Di seguito viene fatta una breve descrizione della struttura generale di ResNet.

4.1.1 ResNet: definizione e struttura

ResNet [7], o rete neurale residua (*Residual Network*), è una tipologia di modello di deep learning simile ad una rete neurale convoluzionale, ma in cui la maggior parte dei layer classici è stata sostituita da dei blocchi residui. Dato un certo input x ed un gruppo di layer sovrapposti che calcola una funzione \mathcal{H} , il corrispondente **blocco residuo** effettua lo stesso calcolo, ma attraverso una riparametrizzazione in modo che:

$$\mathcal{F}(x) = \mathcal{H}(x) - x \quad (7)$$

dove \mathcal{F} è detta *funzione residua*. In questo modo, il blocco residuo ha come uscita:

$$y = \mathcal{F}(x) + x \quad (8)$$

Un esempio di blocco residuo a due layer viene mostrato in Figura 6.

4.1.2 Metriche e scelte implementative

Sempre per avere un confronto con OC-FakeDect1, abbiamo mantenuto le stesse metriche di valutazione delle performance del modello; come conseguenza di ciò, la funzione di perdita utilizzata è quella definita in (3), mentre la misura dell'accuratezza segue la definizione di score di ricostruzione in (5).

Durante il corso degli esperimenti, sono stati utilizzati due differenti versioni di autoencoder variazionale, che per semplicità chiamiamo **ResNetOCVAE1** e **ResNetOCVAE2** e che adesso vengono brevemente descritte evidenziandone le differenze:

- **ResNetOCVAE1:** in questa versione, l'encoder viene implementato utilizzando la classe *resnet18* della libreria *torchvision.models* del framework *Pytorch*, una rete neurale ResNet-18 preaddestrata su ImageNet, un dataset contenente immagini distribuite in 1000 classi differenti; il decoder, invece, ha una struttura a blocchi relativamente semplice, asimmetrica rispetto all'encoder, ma che cerca lo stesso di rispecchiare le operazioni dei blocchi residui di ResNet-18 [8].
- **ResNetOCVAE2:** a differenza di ResNetOCVAE1, l'encoder viene implementato "da zero" senza utilizzare alcun tipo di libreria; anche il decoder ha una struttura differente, meno elementare, ma in questo caso perfettamente simmetrica rispetto a quella dell'encoder [9]

5 Descrizione degli esperimenti condotti

5.1 Preparazione dei dati e fase di addestramento

Prima di procedere alla fase di addestramento dell'autoencoder variazionale, le immagini estratte da FaceForensics++ sono state modificate per "adattarsi" al primo layer di ResNet-18. In particolare, è stata effettuato un ridimensionamento a 224 pixel in altezza e 224 in larghezza, ed una normalizzazione che riflettesse il pre-processing utilizzato per ImageNet. E' importante sottolineare che la normalizzazione di ImageNet è necessaria soltanto per ResNetOCVAE1, ma abbiamo deciso di utilizzarla anche con ResNetOCVAE2 in modo da facilitare il confronto dei due modelli.

Una volta preparate le immagini, esse sono state suddivise in due gruppi:

- il primo relativo al training set, contenente 99700 immagini reali
- il secondo relativo al test set, contenente 250 immagini reali e 250 immagini estratte casualmente da *Deepfakes*

Sia ResNetOCVAE1 che ResNetOCVAE2 sono stati addestrati per 50 epoche, effettuando ogni 10 epoche il salvataggio dei pesi. Come algoritmo di ottimizzazione abbiamo deciso di utilizzare **Adam**, una versione dell'algoritmo di discesa stocastica del gradiente in cui il tasso di apprendimento (*learning rate*) non è fisso, ma varia in base alla frequenza di aggiornamento dei parametri del modello.

5.2 Test effettuati

Ad ogni salvataggio le performance del modello ottenuto sono valutate attraverso l'osservazione di numerosi grafici, che procediamo a descrivere dettagliatamente:

- **Iistogramma dello score di ricostruzione:** è un istogramma che mostra la frequenza con cui lo score di ricostruzione assume determinati valori; durante la fase di test vengono creati due istogrammi separati, il primo relativo agli score ottenuti con le immagini reali ed il secondo relativo agli score ottenuti con i deepfake. In particolare, la frequenza f_i di ciascun intervallo di valori di larghezza w_i non è calcolata tramite conteggio, ma come densità di probabilità, cioè in modo che:

$$\sum_{i=1}^n (f_i * w_i) = 1 \quad (9)$$

Per una migliore comprensione dei risultati, i due istogrammi sono successivamente sovrapposti in un unico grafico.

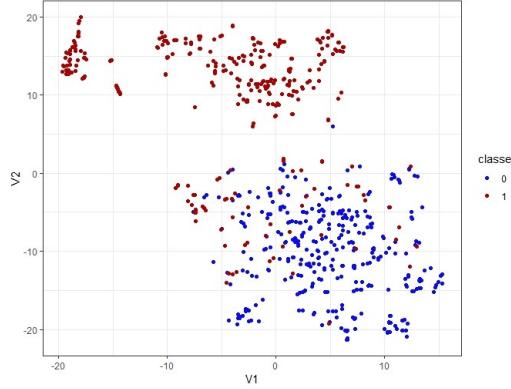


Figura 8: rappresentazione t-SNE mediante grafico di dispersione

- **Curva precisione-richiamo:** prima di spiegare cosa rappresenta questo grafico, è importante ricordare alcuni nozioni teoriche. In un problema di classificazione binaria, dette "positiva" e "negativa" le due classi a cui i dati possono appartenere, si definisce **vero positivo** un dato positivo (cioè etichettato come appartenente alla classe positiva) classificato come positivo, mentre **falso positivo** un dato negativo classificato come positivo (lo stesso ragionamento vale per i veri negativi e per i falsi negativi). Precisione e richiamo sono due metriche molto comuni per la valutazione delle capacità di classificazione di un modello e vengono definite come segue:

$$precision = \frac{\text{veri positivi}}{\text{veri positivi} + \text{falsi positivi}} \quad (10)$$

$$richiamo = \frac{\text{veri positivi}}{\text{veri positivi} + \text{falsi negativi}} \quad (11)$$

dove la prima indica la percentuale di veri positivi sul totale dei dati che il modello ha riconosciuto come positivi, mentre la seconda indica la percentuale di veri positivi sul totale dei dati effettivamente positivi. Invece, viene chiamato **soglia** un parametro che stabilisce il criterio con cui il modello classifica un certo dato e determina la probabilità di quest'ultimo di essere riconosciuto come positivo o negativo. Si definisce dunque **curva precisione-richiamo** (*precision-recall curve*) il grafico che mostra l'andamento di precisione e richiamo al variare della soglia scelta. Idealmente, il risultato desiderato è una curva la cui area sottesa tende a 1, poiché ciò è indice sia di un'alta precisione che di un alto richiamo. Nel caso del nostro problema di anomaly detection, è possibile calcolare precisione e richiamo attraverso gli score di ricostruzione; in particolare, un'immagine x viene classificata come reale, ossia appartenente alla classe positiva, se vale:

$$\text{reconstruction_score}(x, \hat{x}) < s \quad (12)$$

dove s rappresenta la soglia utilizzata, che può assumere valori in $[0, \max_x(\text{reconstruction_score}(x, \hat{x}))]$.

- **Rappresentazione t-SNE dello spazio latente:** t-SNE (*t-distribution Stochastic Neighbor Embedding*) è un algoritmo che consente di ridurre la dimensionalità dei dati per facilitarne la visualizzazione. L'algoritmo si articola in due fasi:

1. prima viene creata una distribuzione normale in modo che a due punti poco distanti nello spazio ad alta dimensionalità sia associato un alto valore di probabilità, viceversa se i punti sono molto distanti;
2. successivamente viene costruita una distribuzione t di Student analoga alla prima per rappresentare i punti nello spazio a dimensionalità ridotta, e viene risolto un problema di ottimizzazione con l'obiettivo di minimizzare la divergenza di Kullback-Leibler tra le due distribuzioni.

Contestualmente al lavoro svolto, t-SNE viene utilizzato per la visualizzazione in due dimensioni dei vettori dello spazio latente prodotti dall'encoder, per cercare di comprendere meglio la capacità dell'autoencoder

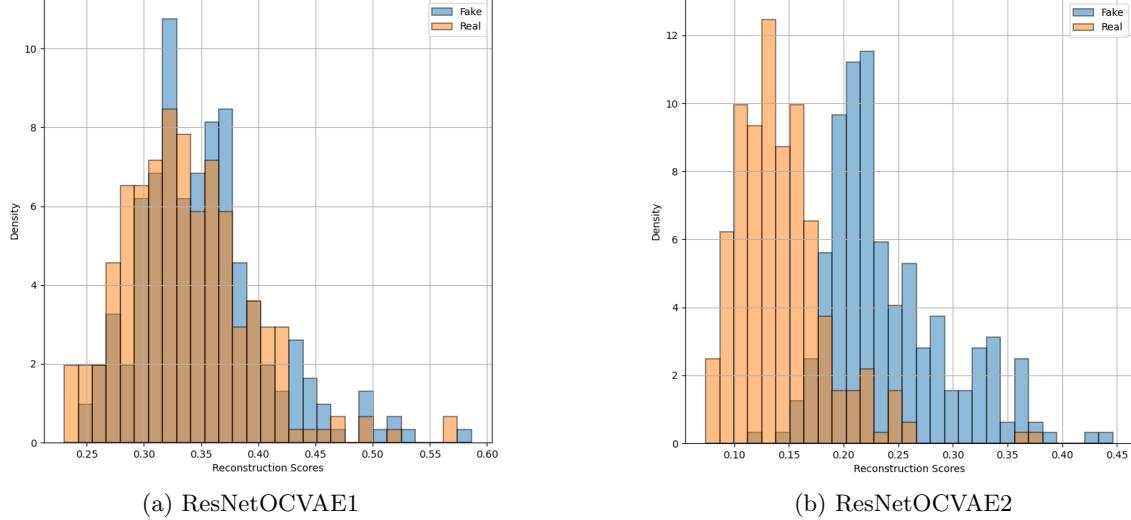


Figura 9: istogrammi degli score di ricostruzione dopo 40 epoche di addestramento

variazionale di campionare adeguatamente i punti dalla distribuzione a posteriori e, a partire da essi, effettuare una ricostruzione delle immagini in ingresso al modello. In particolare, la rappresentazione t-SNE dello spazio latente è mostrata sottoforma di grafico di dispersione (un esempio in Figura 8).

Alla fine della fase di addestramento, abbiamo deciso di eseguire ulteriori test utilizzando la versione del modello, tra quelli salvati, che ha raggiunto i risultati migliori, ossia nel nostro caso quello a cui è associata l'area sottesa dalla curva precisione-richiamo di valore maggiore; questa decisione dipende dal fatto che tale area, come accennato precedentemente, è direttamente proporzionale alla precisione ed al richiamo medio del modello, perciò maggiore sarà il suo valore, migliori saranno le performance e le capacità di classificazione raggiunte.

Nello specifico, questi ultimi test riguardano l'osservazione e l'analisi dell'**errore di ricostruzione** di ogni immagine, definito semplicemente come la differenza tra l'input x e l'output ricostruito \hat{x} :

$$\text{recons_err} = |x - \hat{x}| \quad (13)$$

Tuttavia, non ci siamo limitati a mostrare direttamente il valore degli errori, ma abbiamo voluto includere anche il calcolo della loro **trasformata di Fourier discreta** (DFT, *Discrete Fourier Transform*) su tre dimensioni, per poter effettuare un'analisi nel dominio delle frequenze:

$$\text{recons_err}_{c'h'w'} = \mathcal{F}(\text{recons_err}_{chw}) = \sum_{c=0}^{C-1} \sum_{h=0}^{H-1} \sum_{w=0}^{W-1} \text{recons_err}_{chw} * e^{-2\pi j (\frac{c'c}{C} + \frac{h'h}{H} + \frac{w'w}{W})} \quad (14)$$

per $c' = 0, \dots, C - 1$, $h' = 0, \dots, H - 1$, $w' = 0, \dots, W - 1$

La trasformata viene visualizzata attraverso lo **spettro di ampiezza**, ossia $|\text{recons.err}|$, centrato sulla frequenza zero.

6 Presentazione ed analisi dei risultati sperimentali

In questa sezione vengono mostrati i risultati degli esperimenti descritti nella sezione 5.2; per ogni tipologia di test, ResNetOCVAE1 e ResNetOCVAE2 sono messi a confronto, in modo da evidenziare le differenze tra i due modelli e cercare di comprendere quale dei due è il migliore. In particolare, i grafici presentati sono quelli relativi alle versioni dei due autoencoder variazionali ottenute dopo 40 epoche di addestramento poiché, in entrambi i casi, le performance migliori sono state raggiunte dopo tale quantità di tempo (sulle 50 epoche totali).

6.1 Istogramma degli score di ricostruzione

Osservando la Figura 9, gli istogrammi degli score di ricostruzione di ResNetOCVAE1 (9a) e ResNetOCVAE2 (9b) presentano delle differenze significative:

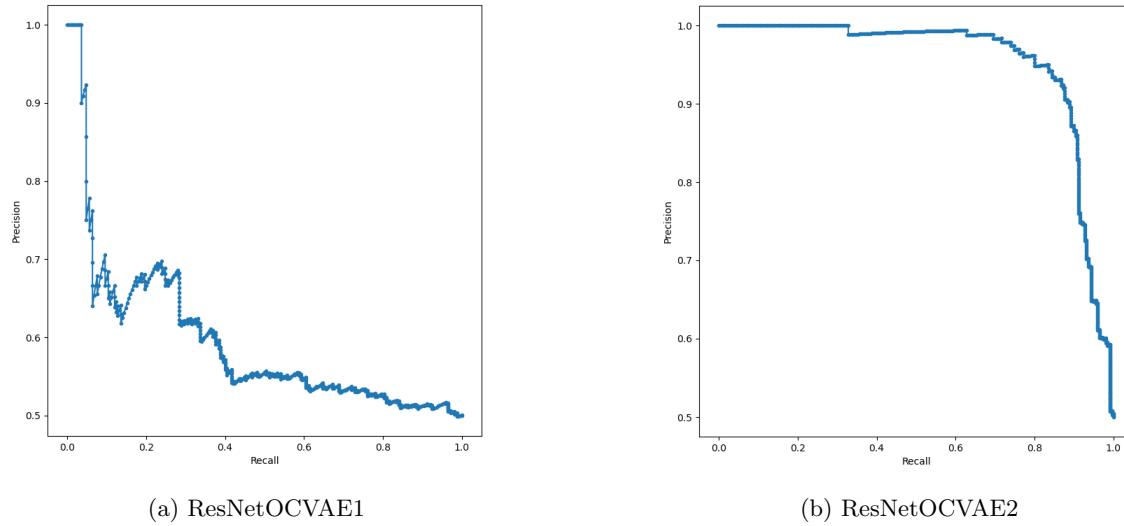


Figura 10: curve precisione-richiamo dopo 40 epoche di addestramento

- la prima, meno marcata, riguarda la distribuzione dei valori che gli score di ricostruzione assumono; essi si concentrano intorno a valori di 0.30-0.40 nel caso di ResNetOCVAE1 ed intorno a 0.10-0.25 nel caso di ResNetOCVAE2, con dei picchi relativamente isolati di valori 0.60 e 0.45 rispettivamente
 - la seconda riguarda la percentuale di istogramma relativo alle immagini reali che si sovrappone con quello relativo ai deepfake; infatti, nel caso di ResNetOCVAE1, i due histogrammi sono quasi del tutto sovrapposti, ad indicare che il modello non riesce a vedere le immagini artificiali come anomalie nel dataset delle immagini reali, nonostante riesca ad effettuare una discreta ricostruzione per entrambe le tipologie di dati. Nel caso di ResNetOCVAE2, invece, la sovrapposizione tra i due grafici è decisamente inferiore, limitata solamente ai valori dello score nell'intervallo 0.15-0.25; ciò ha come conseguenza il fatto che a sinistra della zona sovrapposta si concentrano maggiormente gli score di ricostruzione relativi alle immagini reali, i quali assumono quindi valori molto piccoli, viceversa nel caso dei deepfake. Dunque, il secondo modello mostra delle performance nettamente superiori rispetto al primo nella task di anomaly detection, riuscendo ad ottenere comunque degli score di ricostruzione notevoli sia nel caso di dati non anomali che nel caso di quelli anomali.

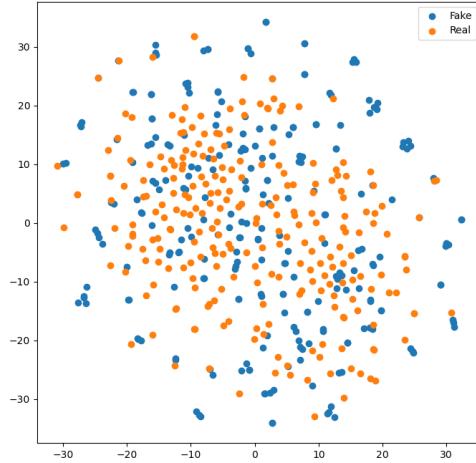
6.2 Curva precisione-richiamo

Le curve precisione-richiamo in Figura 10 mostrano una situazione analoga alla precedente: nella prima, relativa a ResNetOCVAE1, il valore della precisione cala a picco quasi immediatamente; ciò significa che il modello riconosce correttamente un’immagine reale ricostruita quando il numero di esempi è molto basso (cioè quando il richiamo è piccolo), ma commette un’elevata quantità di errori nel caso in cui tale numero viene incrementato. Al contrario, nel caso di ResNetOCVAE2 i valori di precisione e richiamo si mantengono moderatamente alti indipendentemente dal valore che la soglia assume, e questo lo si nota anche osservando la differenza marcata tra l’area sottesa dalla curva in Figura 10a e quella in Figura 10b. Anche in questo caso, quindi, ResNetOCVAE2 ottiene dei risultati migliori rispetto a ResNetOCVAE1.

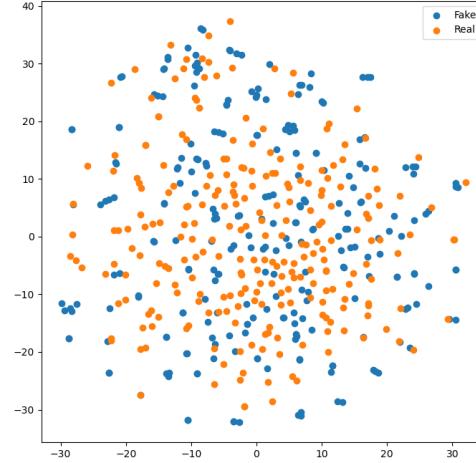
6.3 Rappresentazione t-SNE dello spazio latente

A differenza degli esperimenti precedenti, i grafici di dispersione relativi alla rappresentazione t-SNE dello spazio latente non sono visibili differenze significative tra ResNetOCVAE1 e ResNetOCVAE2; la Figura 11 mette in evidenza due elementi comuni ai grafici di dispersione relativi ai due modelli:

- non è presente nello spazio una netta separazione tra punti arancioni, associati alle immagini reali, e punti blu, associati ai deepfake: ciò fa comprendere che per entrambi i modelli, in termini di caratteristiche nello spazio latente, un’immagine reale ed un deepfake sono visti come dati appartenenti ad una stessa categoria.



(a) ResNetOCVAE1



(b) ResNetOCVAE2

Figura 11: rappresentazioni t-SNE dello spazio latente dopo 40 epoche di addestramento

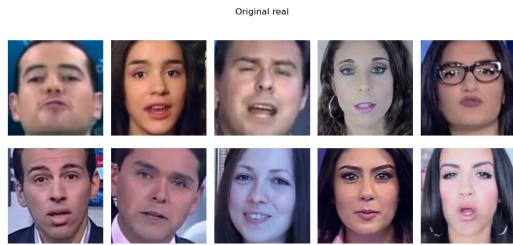


Figura 12: campione di immagini reali

Questo comportamento è da una parte comprensibile, perché entrambe le tipologie di immagine hanno come soggetto principale un volto umano

- tutti i punti tendono a concentrarsi all'interno di un'area di forma circolare: questa caratteristica, invece, evidenzia il tentativo da parte di ResNetOCVAE1 e ResNetOCVAE2 di rendere la scelta dei vettori dello spazio latente un'operazione di campionamento da una determinata distribuzione (nel nostro caso una distribuzione normale standard), ovvero uno degli obiettivi che normalmente si cerca di raggiungere attraverso l'addestramento di un autoencoder variazionale

6.4 Errori di ricostruzione

In questa sezione vengono presentati gli esiti dei test svolti con ResNetOCVAE1 e ResNetOCVAE2 circa l'errore di ricostruzione delle immagini; però, a differenza degli esperimenti precedenti, abbiamo ritenuto ragionevole



Figura 13: campione di immagini deepfake

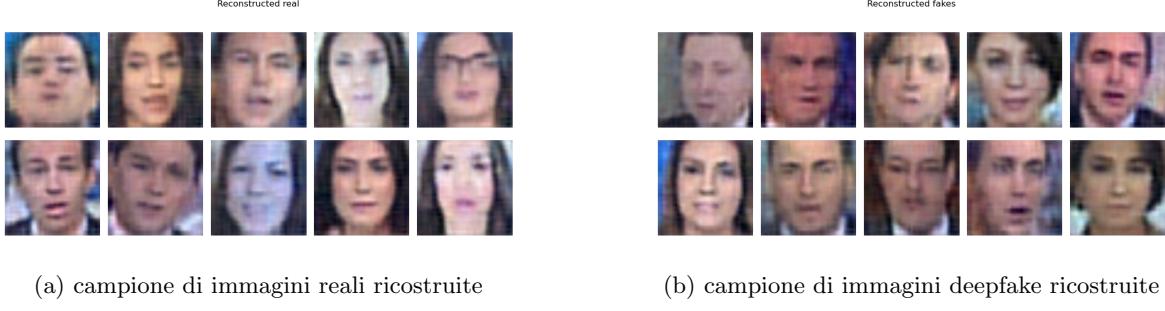


Figura 14: ricostruzione di immagini reali e deepfake con ResNetOCVAE1

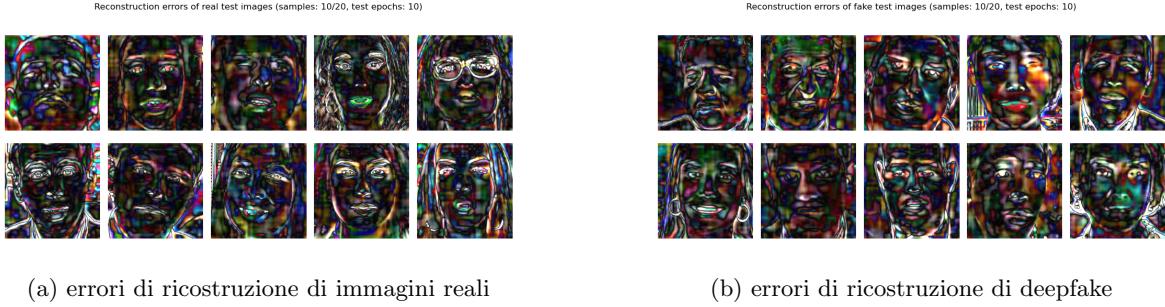


Figura 15: errori di ricostruzione di immagini reali e deepfake con ResNetOCVAE1

effettuare un confronto tra risultati ottenuti con immagini reali e con immagini deepfake, piuttosto che un paragone diretto tra i due autoencoder variazionali utilizzati. Questa decisione permette di comprendere in modo migliore sotto quali aspetti, qualora ci fossero, un modello differisce dall’altro, vista la natura di questi ultimi esperimenti condotti.

In Figura 12 e 13 sono mostrati, rispettivamente, il campione di 10 immagini reali e quello di 10 deepfake in base ai quali sono stati calcolati gli errori di ricostruzione.

6.4.1 ResNetOCVAE1

Prima di discutere degli errori di ricostruzione, mostriamo le ricostruzioni dei campioni di immagini reali e di deepfake citati in precedenza, rispettivamente in Figura 14a e 14b; è possibile notare che le immagini, ottenute con ResNetOCVAE1, presentano correttamente le caratteristiche ed i tratti generali dei volti originali, ma appaiono quasi ”sfocate”, indice del fatto che il modello fallisce nel ricostruire la maggior parte dei dettagli più specifici. Inoltre, non sembra esserci una differenza significativa tra l’aspetto delle immagini reali e quello delle immagini artificiali.

Con l’osservazione degli errori di ricostruzione nelle Figure 15a e 15b viene confermata l’incapacità del modello di essere discriminativo tra dati normali ed anomalie, così come viene messa ulteriormente in evidenza la difficoltà

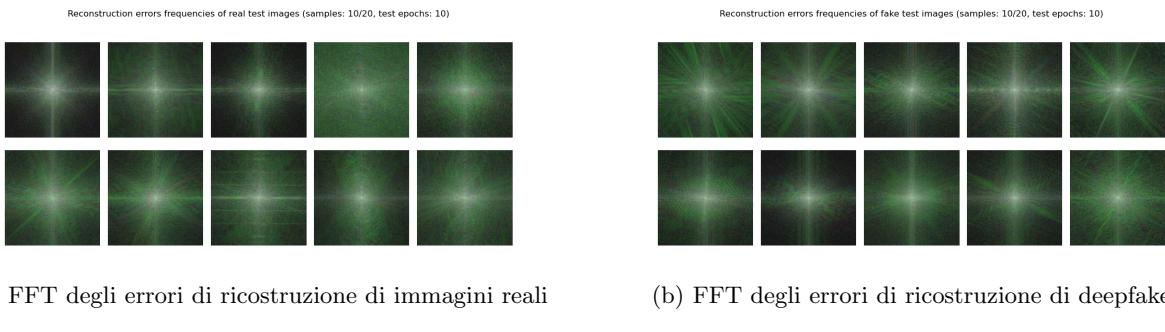


Figura 16: errori di ricostruzione di immagini reali e deepfake con ResNetOCVAE1 nel dominio della frequenza

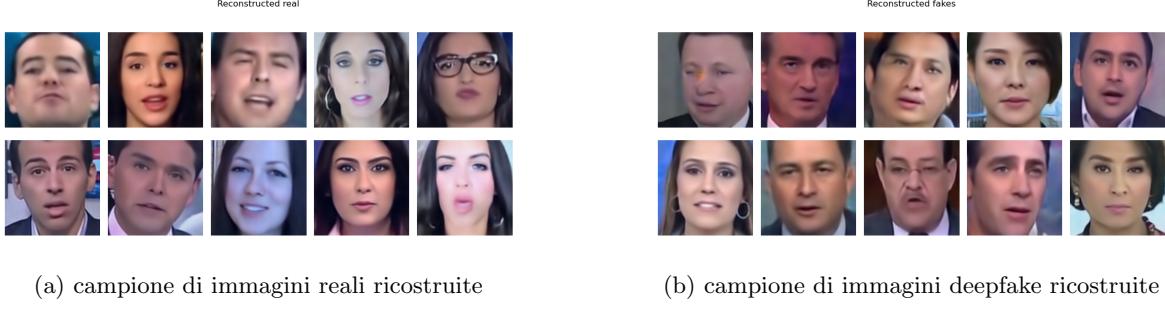


Figura 17: ricostruzione di immagini reali e deepfake con ResNetOCVAE2

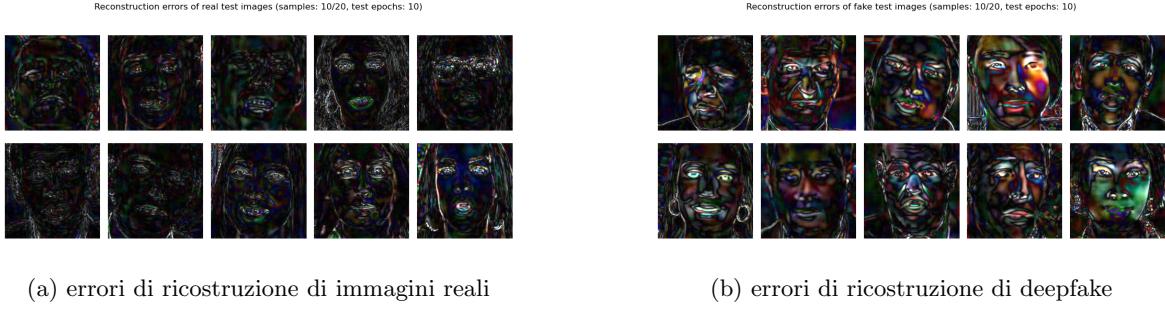


Figura 18: errori di ricostruzione di immagini reali e deepfake con ResNetOCVAE2

di quest'ultimo nel generare immagini simili a quelle originali; infatti, per entrambe le tipologie di immagini sono presenti una quantità non trascurabile di pixel colorati, soprattutto nella zona corrispondente ai contorni dei volti, in un esperimento in cui il risultato desiderato prevede la preponderanza, nelle immagini corrispondenti agli errori, di pixel di colore nero (si desidera cioè minimizzare l'errore di ricostruzione). Questi residui colorati rappresentano proprio gli elementi dettagliati dei volti presenti nelle immagini originali che il modello non è riuscito a ricostruire.

L'analisi degli errori nel dominio delle frequenze è più complessa rispetto alle precedenti; infatti, come si può osservare dalle Figure 16a e 16b, non sembra esserci una differenza significativa tra immagini reali ed artificiali. In entrambi i casi la distribuzione delle frequenze non è ben definita, con picchi che a volte interessano principalmente le alte frequenze, mentre altre volte sono riscontrabili anche in quelle basse. Questo comportamento indica che gli errori di ricostruzione commessi da ResNetOCVAE1 non sono sistematici, ma casuali e poco prevedibili, un'ulteriore conferma delle scarse capacità di questo modello per la task richiesta. Tuttavia, per poter affermare con certezza quanto detto sarebbero necessari ulteriori approfondimenti, per esempio effettuando un analisi dello spettro di potenza degli errori di ricostruzione.

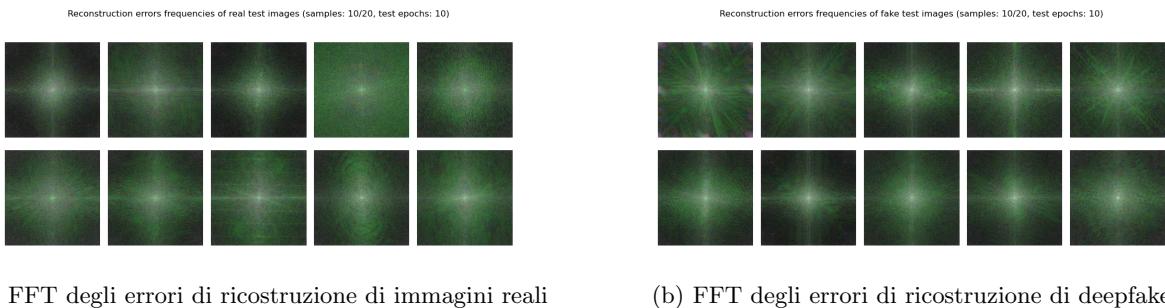


Figura 19: errori di ricostruzione di immagini reali e deepfake con ResNetOCVAE2 nel dominio della frequenza

	soglia (80 percentile)	precisione	richiamo	accuratezza
OC-AE (baseline)	0.017	0.492	0.492	0.492
OC-FakeDect1	0.012	0.860	0.864	0.862
ResNetOCVAE1	0.383	0.519	0.828	0.528
ResNetOCVAE2	0.242	0.604	0.964	0.664

Tabella 1: Confronto delle performance tra un autoencoder per la classificazione ad una classe (OC-AE), OC-FakeDect1, ResNetOCVAE1 e ResNetOCVAE2 sulla forgery *Deepfakes* di FaceForensics++. Da notare che precisione e richiamo sono calcolati solo rispetto alla classe positiva (immagini reali)

	soglia (80 percentile)	soglia (validation set)	precisione	richiamo	accuratezza
OC-FakeDect1	0.012		0.860	0.864	0.862
ResNetOCVAE2		0.180	0.910	0.888	0.900

Tabella 2: Confronto tra OC-FakeDect1 e ResNetOCVAE2 utilizzando la soglia calcolata tramite validation set

6.4.2 ResNetOCVAE2

Basta osservare le immagini ricostruite da ResNetOCVAE2 in Figura 17 per comprendere che questo modello ottiene dei risultati decisamente superiori rispetto a ResNetOCVAE1. Infatti, non solo le immagini contengono la maggior parte dei dettagli facciali, ma questa volta il modello ha un comportamento discriminativo piuttosto marcato, riuscendo a ricostruire con fedeltà le immagini reali ed allo stesso tempo commettendo diversi errori nel ricostruire deepfake (come si può vedere bene in Figura 17b).

Anche gli errori di ricostruzione in Figura 18a e 18b mostrano delle differenze significative tra immagini reali e deepfake. In particolare, gli errori relativi alle immagini reali sono quasi del tutto assenti, e ciò lo si nota dalla quantità ridotta di residui rispetto ai risultati ottenuti con ResNetOCVAE1; nel caso dei deepfake, invece, la presenza di residui è ben più evidente, elemento che indica la maggiore difficoltà di ricostruzione che il modello ha, obiettivo che volevamo raggiungere.

Come per ResNetOCVAE1, l’analisi degli errori nel dominio delle frequenze è inconcludente e non ci permette di sostenere nessuna posizione di pensiero con certezza; tuttavia, è possibile notare una differenza più marcata tra la distribuzione delle frequenze nelle immagini in Figura 19a, uniforme con pochi picchi sulle basse frequenze, e quella nelle immagini in Figura 19b, non ben definita e con una grande quantità di picchi sia nelle basse che nelle alte frequenze.

7 Confronto con OC-FakeDect1

E’ importante effettuare un confronto anche con OC-FakeDect1 per comprendere se ResNetOCVAE1 e ResNetOCVAE2 rappresentano un miglioramento rispetto a quello che è il punto di partenza del nostro lavoro. In particolare, vengono confrontati precisione, richiamo ed accuratezza relativi agli score di ricostruzione ottenuti con ciascuno dei tre modelli. Come soglia viene scelta la stessa utilizzata in [5], ossia l’**80 percentile** della distribuzione degli score.

Nella Tabella 1 sono riportati i valori ottenuti con OC-FakeDect1, ResNetOCVAE1, ResNetOCVAE2, e con un autoencoder semplice (OC-AE), ”baseline” scelta dagli autori di [5]; nonostante i VAE da noi sviluppati abbiano ottenuto precisione, richiamo ed accuratezza maggiori di quelli della baseline, nessuno di essi rappresenta un vero e proprio miglioramento rispetto a OC-FakeDect1; l’unica eccezione è il richiamo di ResNetOCVAE2, con un valore che supera di circa il 10% quello relativo al modello degli autori.

Tuttavia, dall’osservazione della curva di precisione-richiamo in Figura 10b sembra che con ResNetOCVAE2 si possa aumentare notevolmente la precisione, diminuendo il relativo richiamo. Per farlo, abbiamo deciso di utilizzare un set di validazione contenente 250 immagini reali e 250 deepfake, con il quale viene calcolato l’**F1 score** degli score di ricostruzione, definito come la media armonica di precisione e richiamo:

$$F1score = 2 * \frac{precisione * richiamo}{precisione + richiamo} \quad (15)$$

Questa metrica di accuratezza, infatti, tiene conto sia della precisione che del richiamo in modo bilanciato, così che ad un F1 score elevato corrispondano dei buoni valori per entrambi i parametri. In particolare, tramite il set

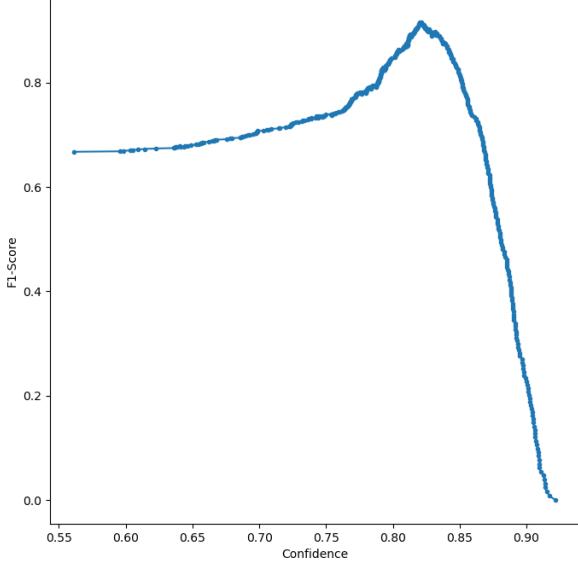


Figura 20: F1 score degli score di ricostruzione al variare della soglia

di validazione vengono determinati gli F1 score per ogni soglia (Figura 20), e viene estratta quella corrispondente all’F1 score massimo; con questa soglia sono calcolati precisione, richiamo ed accuratezza sul set di test.

Nella Tabella 2 sono confrontati i risultati ottenuti da ResNetOCVAE2 con questo secondo approccio e quelli ottenuti da OC-FakeDect1 con l’approccio originale. Come ci aspettavamo, la nuova soglia permette a ResNetOCVAE2 di ottenere precisione, richiamo ed accuratezza superiori rispetto a quelli raggiunti con OC-FakeDect1.

Per un miglioramento di questo livello, l’utilizzo di un set di validazione disgiunto dal set di test per il calcolo della soglia è cruciale; infatti, contestualmente ad una task di deepfake detection, lavorando esclusivamente con i dati di test il rischio è quello di perdere in generalizzazione sul riconoscimento dei dati artificiali, soprattutto nella situazione in cui nascono nuove tecniche per la manipolazione delle immagini, le quali richiedono un aggiornamento del modello.

8 Conclusioni e lavori futuri

In questo lavoro sul riconoscimento dei deepfake mediante anomaly detection con classificazione ad una classe, abbiamo presentato due autoencoder variazionali, come alternativa al modello già proposto in [5], ovvero OC-FakeDect1; il primo, il VAE asimmetrico ResNetOCVAE1, non ha ottenuto in generale dei buoni risultati, superando per prestazioni solo quelle di un autoencoder semplice; il secondo, il VAE simmetrico ResNetOCVAE2, al contrario, ha ottenuto dei discreti risultati utilizzando una soglia scelta con lo stesso criterio di [5], e ha ottenuto degli ottimi risultati utilizzando la soglia selezionata tramite set di validazione. Perciò, possiamo dire che con qualche modifica all’approccio di scelta della soglia degli score di ricostruzione, è possibile ottenere dei risultati migliori rispetto allo stato dell’arte anche con degli autoencoder variazionali di complessità elevata, a patto che abbiano una struttura simmetrica.

In futuro potrebbe essere interessante ripetere gli stessi esperimenti nelle stesse condizioni, ma utilizzando un VAE basato su OC-FakeDect2 invece di OC-FakeDect1, per comprendere se un ulteriore aumento di complessità porta a dei miglioramenti nelle performance del modello; un altro spunto da non escludere potrebbe essere l’approfondimento dell’analisi degli errori di ricostruzione nel dominio delle frequenze, come accennato alla fine della sezione 6.4.1, in modo tale da poter rafforzare, o eventualmente smentire, le ipotesi formulate sulle possibili interpretazioni dei risultati ottenuti. Infine, reputiamo significativo provare ad espandere i set di test e di addestramento con le immagini appartenenti alle altre forgery di FaceForensics++, per comprendere se anche questa aggiunta possa influenzare i risultati dei test condotti.

9 Ringraziamenti

Non possono mancare dei ringraziamenti speciali alla mia famiglia, in particolare ai miei genitori e alle mie nonne, per avermi sempre sostenuto e per avermi fatto diventare quello che sono oggi. Un altro ringraziamento va ai miei amici più stretti, per il supporto che anche da loro ho sempre ricevuto. Questo lavoro lo vorrei dedicare soprattutto ai miei nonni che purtroppo non ci sono più, ma che sarebbero stati fieri di tutto il percorso che ho fatto fino a questo punto.

10 Bibliografia

- [1] https://www.repubblica.it/spettacoli/people/2024/01/28/news/taylor_swift_x_blocca_le_ricerche_dopo_le_false_immagini_di_nudo_create_con_intelligenza_artificiale-421998674/
- [2] <https://www.wired.it/article/deepfake-joe-biden-elezioni-stati-uniti/>
- [3] C. Han et al., "GAN-based synthetic brain MR image generation," 2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018), Washington, DC, USA, 2018, pp. 734-738
- [4] Chih-Chung Hsu, Yi-Xiu Zhuang, and Chia-Yen Lee. Deep fake image detection based on pairwise learning. *Applied Sciences*, 10(1):370, 2020
- [5] Hasam Khalid, Simon S. Woo; OC-FakeDect: Classifying Deepfakes Using One-Class Variational Autoencoder, proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, 2020, pp. 656-657
- [6] Rossler, A., Cozzolino, D., Verdoliva, L., Riess, C., Thies, J., & Nießner, M. (2019). Faceforensics++: Learning to detect manipulated facial images. In Proceedings of the IEEE/CVF international conference on computer vision (pp. 1-11)
- [7] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778)
- [8] <https://github.com/hsinyilin19/ResNetVAE/tree/master>
- [9] <https://github.com/eleannavali/resnet-18-autoencoder/tree/main>