

**METODI E MODELLI PER LA SICUREZZA DELLE APPLICAZIONI**

**Rilevamento audio deepfake**

*Docenti: Prof. Donato Impedovo*

*Co-relatore: Dott. Francesco Castro*

*Studente: Gabriele Nigro*

Sommario

[1. Introduzione 3](#_heading=h.gjdgxs)

[2. Stato dell’arte 3](#_heading=h.1fob9te)

[3. Dataset 3](#_heading=h.3znysh7)

[4. Metodi 3](#_heading=h.2et92p0)

[5. Sperimentazione 3](#_heading=h.tyjcwt)

[6. Risultati 3](#_heading=h.3dy6vkm)

[7. Conclusioni 3](#_heading=h.1t3h5sf)

[Riferimenti 3](#_heading=h.2s8eyo1)

[Bibliografia 4](#_heading=h.17dp8vu)

# **Introduzione**

Per “deepfake” si intendono tutti quei contenuti multimediali quali audio, foto, video, generati con l’intelligenza artificiale, tramite l’utilizzo di reti neurali artificiali addestrate allo scopo.

Il primo progetto di creazione di deepfake definibili tali risale al 1997 con il Video Rewrite Program[1] , lavoro seminale ad opera di allora studenti e dottorandi il cui obiettivo, raggiunto con risultati ottimi per l’epoca, era di riadattare il labiale in un video per farlo combaciare con una differente traccia audio. Fu il primo sistema ad automatizzare questo processo usando l’apprendimento profondo

Il progresso che si ha avuto nel campo dell’intelligenza artificiale negli ultimi 2-3 decenni ha fatto sì che la creazione di contenuti deepfake sia oggi più facile che mai ed è oggi disponibile al grande pubblico senza richiedere competenze tecniche, grazie a modelli quali Stable Diffusion resi disponibili anche online. Questi richiedono solo un prompt testuale per generarne l’immagine corrispondente. Ma la creazione di reti neurali avanzate risulta più semplice anche agli addetti ai lavori tramite l’uso di API di alto livello come Keras. Inoltre la letteratura su questo argomento è in costante crescita da una decina di anni a questa parte

L’argomento deepfake è molto spinoso, tant’è che non viene studiato solo da un punto di vista informatico ma anche da quello etico. I campi d’applicazione dei deepfake sono generalmente nell’ambito artistico. Possono essere usati per creare cloni digitali di attori o per farne il cosiddetto de-aging, “ringiovanire” digitalmente un attore. Possono essere usati per il doppiaggio o, come accennato prima, creare arte digitale facilmente. Dall’altro lato il loro utilizzo è associato alle frodi, alla disinformazione soprattutto politica che viene amplificata dai social media, ai ricatti e al revenge porn.

**Reti neurali**

Una rete neurale feedforward implementa una funzione di predizione f(x) data dalla composizione di N funzioni fn(fn-1(fn-2(….(f1(x)))). Ciascuna di queste mappa da un certo vettore di valori di input ad un certo vettore di valori di output ed è chiamata “strato” della rete: n funzioni vuol dire rete neurale di n strati nascosti. Ogni componente del vettore è detto unità o “neurone”. Il deep learning consiste quindi di modelli composti da tanti strati il cui compito è imparare a rappresentare i dati con molteplici livelli di astrazione. Questo tipo di rete neurale si dice “feedforward” perché il flusso di informazioni viaggia a senso unico dallo strato di input a quello di output, senza cicli come possono comparire invece in una rete ricorrente.

Ogni strato è una funzione lineare di parametri da apprendere di ogni output dato l’input, seguito da una funzione di attivazione nonlineare, come ad esempio la Rectified Linear Unit (Relu(x) = max(0,x)) e differenziabile, affinchè la discesa del gradiente funzioni. Lo strato finale della rete, quello deputato alla predizione finale, presenta tipicamente una funzione di attivazione differente, la cui scelta dipende dall’output previsto e la loss function da minimizzare. Nel caso di questo progetto, per una classificazione binaria il cui output è un valore 0 o 1 ed una log loss binaria da minimizzare, la funzione di attivazione finale sarà una sigmoide (sigmoid(x) = 1/1+e^-x)), che in output restituisce un valore tra 0 e 1.

La task di questo progetto è una di apprendimento supervisionato, dunque si dispone di un insieme di dati di addestramento sotto forma di coppie input-output, usando i quali la rete cerca di inferire la relazione tra i due, allo scopo di imparare a produrre un output corretto avendo a disposizione solo un input, la rete deve cioè imparare a generalizzare su dati non visti prima. Con un algoritmo di backpropagation viene calcolato il gradiente della loss function rispetto ai pesi della rete e questi ultimi vengono aggiornati con metodi come quello della discesa del gradiente

La peculiarità di una rete neurale convoluzionale (Convolutional Neural Network), motivo per il quale sono indicate per la classificazione di immagini, è che riesce a tenere in considerazione le dipendenze spazio-temporali dei pixel, ovvero la nozione secondo cui pixel spazialmente vicini sono correlati e frame contigui di un video presentano poche differenze. Una rete convoluzionale applica dei filtri chiamati kernel su piccole parti di un immagine, operatori lineari, tipicamente prodotto scalare. Il kernel viene fatto “passare” su un immagine e viene calcolato il prodotto scalare tra un valore del kernel e il corrispondente valore nell’immagine. Il kernel viene poi fatto “strisciare” lungo l’immagine, spostato di un certo numero di pixel chiamato “stride”. Una volta terminato il processo di convoluzione, si ha per risultato una feature map, tipicamente di dimensione minore rispetto all’input poiché il kernel può “uscire fuori dai bordi dell’immagine”, gli elementi ai bordi vengono rimossi. Comune soluzione a questo problema è lo zero padding, aggiungere zeri ai bordi dell’immagine. Ciò che contraddistingue queste reti è il principio di località: i valori sono una funzione dei valori nelle posizioni vicine: un kernel 5x5 usa due pixel contigui in ogni direzione per calcolare il valore di un solo pixel di output. Inoltre c’è il principio della condivisione dei parametri: per un singolo kernel vengono usati stessi parametri lungo tutta l’immagine.

Quando si costruisce un modello convoluzionale si è soliti aggiungere alla rete anche degli strati di “pooling” che servono da aggregatori di valori: piccoli kernel posizionati sull’immagine danno come output il valore più grande tra quelli sotto il kernel. Lo strato di pooling compie quindi un’operazione di downsampling, di riduzione della dimensionalità della feature map e questo aiuta con l’overfitting e migliora l’efficienza computazionale.

Di norma come ultimi strati di un modello convoluzionale si hanno i cosiddetti fully connected layer (FC). Questi sono degli strati detti “densi” poiché ogni unità è collegata a tutte le unità dello strato precedente. Se le convoluzioni lavorano ad un livello locale, questi strati sono globali, c’è flusso di informazione proveniente da ogni neurone dello strato precedente che va ad ogni classe di output.  
I dati che devono entrare in uno strato fully connected devono però prima essere “appiattiti” in un vettore unidimensionale da uno strato delegato a questo compito. A questo punto, l’ultimo strato della rete è un FC con un numero di neuroni pari al numero di classi possibili.

Quello che invece contraddistingue le reti residuali, che sono un tipo particolare di CNN, sono le cosiddette skip connections. Le reti neurali profonde soffrono di degradazione delle performance all’aumentare della profondità, il cui motivo non è ben chiaro: è un’osservazione empirica, si sa solo che non è un problema di overfitting. La soluzione consiste nelle skip connections: se un certo numero di layer producono alla fine un output h(x), riparametrizziamoli in modo che rappresentino invece una funzione F(x) = h(x) -x. L’output sarà poi F(x) + x con quest’ultima somma che viene implementata tramite una skip connection. La skip connection esegue una funzione identità che collega la x data in input precedentemente con l’output F(x). L’input cioè viene anche “conservato”, salta i layer e viene sommato ad F(x). Essendo una funzione identità non aggiunge parametri al modello e non aumenta la complessità computazionale. Il paper di He et al. [2] che introdusse in concetto di rete residuale in letteratura mostra come queste connessioni residuali risolvano il problema di degradazione delle performance delle reti neurali.

# **Stato dell’arte**

Come paper di riferimento per lo studio dello stato dell’arte è stato selezionata la comprehensive review “*Deepfakes generation and detection: state-of-the-art, open challenges, countermeasures, and way forward*”[3], un paper dettagliato sullo stato dell’arte sia sulla generazione sia sulla rilevazione di contenuti deepfake.

Tra i lavori sulla rilevazione deepfake citati all’interno di questo paper è citato anche il lavoro dei creatori del dataset utilizzato per questo progetto: In Reimao et al. [4] si introduce il dataset e si mostrano i risultati ottenuti con diverse reti neurali addestrate su diversi tipi di feature. Viene utilizzata la versione for-2sec e dalle tracce audio vengono estratti gli MFCC, CQT, Mel Spectrograms e STFT. Su queste feature sono stati addestrate diverse reti neurali, dalle più semplici reti a 3 strati convoluzionali fino a modelli come ResNet o VGG19. Prendendo in considerazione i risultati ottenuti dalla Resnet, hanno raggiunto un 84% sugli spettrogrammi e 88% sugli MFCC.

In “Wang et al.” [5] si introduce un framework chiamato DeepSonar, basato su un sistema di speech recognition dove viene catturato il comportamento dei neuroni attivati di una rete neurale sulle feature audio di input. Questo comportamento diventa un vettore in input ad un classificatore binario che giudica se l’audio è deepfake o meno. In media sono riusciti ad ottenere una accuracy e un F1 score del 98%

Altri paper consultati per confrontare le performance ottenute in questo progetto, sempre rimanendo sul dataset Fake-Or-Real, sono i seguenti:

* Papastergiopoulos et al. [6] hanno addestrato la rete convoluzionale VGG16 su una serie di spettrogrammi, ottenendo un’accuracy del 68% e un F1 score di 64.5% sullo stesso tipo di spettrogramma usato in questo progetto
* Hamza et al. [7] si è proposto di usare un ensemble di feature che include anche gli MFCC per addestrare una rete neurale ricorrente Long Short Term Memory e una rete convoluzionale VGG16, ottenendo rispettivamente accuracy del 91% e 93%
* In Khochare et al. [8] vengono proposte due architetture di reti neurali: una Temporal Convolutional Network [9] e una Spatial Transformer Network [10]. La prima è una rete convoluzionale unidimensionale composta da tre blocchi TCN che applicano convoluzione, leaky ReLU, dropout spaziale (che si differenzia dal normale dropout perché azzera completamente randomicamente ma sceglie specifiche dimensione nella feature map da azzerare) e una connessione residuale. L’input è passato a tre blocchi TCN di kernel a dimensione crescente, poi viene applicato pooling e i tre output vengono concatenati , normalizzati e passati attraverso un layer di dropout standard, per poi rilasciare la classificazione finale. La rete STN invece è una rete convoluzionale bidimensionale che effettua trasformazioni spaziali sulle feature map per rendere la rete più invariante a modifiche spaziali dell’input quali ad esempio rotazioni o traslazioni. Queste due reti ottengono rispettivamente test accuracy di 92% e 80%

# **Dataset**

Il dataset utilizzato per questo progetto è il Fake-or-Real [4] , dataset creato da ricercatori dell’Università di York. Questo dataset è una collezione di quasi 200 mila tracce audio provenienti da esseri umani o generate artificialmente. Le tracce audio genuine provengono a loro volta da ulteriori database pubblici mentre le tracce audio deepfake sono state generate da sistemi Text To Speech stato dell’arte, quali WaveNet [11] e Deep Voice 3 [12] .

Il dataset è diviso in quattro versioni

* For-original è la versione originale, con tracce audio collezionate dalle varie fonti e senza alcun tipo di modifica o preprocessing
* For-2sec è basato sull’*original* ma gli audio sono troncati ad una durata di due secondi, quelli più corti sono stati scartati
* For-norm contiene gli stessi file dell’*original* ma è bilanciato in termini di sesso dello speaker e di numerosità dei campioni delle classi. Inoltre è normalizzato in termini di sample rate (tutte le tracce audio hanno una frequenza di campionamento di 16kHz), volume (0 dBFS) e numero di canali (le tracce audio sono tutte in mono)
* For-rerec è una versione ri-registrata del *2sec* per simulare uno scenario di attacco tramite canale vocale, come ad esempio una telefonata a fin di truffa

Il dataset risulta già diviso in cartelle di testing, training e validation ed è così suddiviso: 77.73% training, 15.58% validation, 6.68% testing. Le tracce audio hanno una lunghezza variabile dagli 0.1 secondi fino anche a tracce audio di 30 secondi.

Si è scelto di utilizzare la versione for-norm per la praticità di avere a disposizione un dataset già bilanciato. Questa versione contiene circa 54000 tracce audio di training, 10800 di validazione e 4600 di test

# **Metodi**

La versione del dataset utilizzato si presenta già bilanciata e normalizzata, per cui non si sono reputate necessarie operazioni di preprocessing sulle tracce audio, se non un filtro in base alla durata della stessa, che viene descritto nella prossima sezione

Per lo scopo di questo progetto si sono implementate ed addestrate due tipi di reti convoluzionali. La prima, che verrà chiamata “standard” d’ora in poi, consiste in una rete convoluzionale composta da 3 blocchi convoluzionali e 3 layer fully connected. Ogni blocco convoluzionale applica in sequenza una convoluzione, un dropout del 10%, una batch normalization e un downsampling attraverso un layer max pooling.  
Dopo i 3 blocchi convoluzionali l’input viene appiattito e si passa tra due layer FC rispettivamente di 64 e 32 unità, in uscita dai quali viene applicato un dropout del 30%. L’output layer è un layer denso a 1 unità, essendo la task una classificazione binaria, con funzione di attivazione sigmoide.

La Resnet implementata è stata largamente ispirata dal paper originale [2]e da implementazioni in altri progetti e paper. È composta da quattro blocchi residuali, un layer FC a 512 unità e un output layer identico a quello per la Cnn standard. Un blocco residuale applica per due volte in ordine una batch normalization, una ReLU, una convoluzione per poi applicare due ulteriori convoluzioni consecutive e infine implementare la skip connection. Il numero di unità nelle convoluzioni è crescente di blocco in blocco: 64 -> 128-> 256 -> 512. L’ordine delle operazioni è stato scelto sulla base dei risultati presentati nel paper di He et al., figura 4e [13].  
 Entrambe le reti utilizzano come ottimizzatore Adam.

# **Sperimentazione**

Ad inizio progetto sono stati eseguiti una serie di esperimenti che sono stati successivamente non considerati in virtù del miglioramento della qualità del codice e del progresso dello studio sulla materia. Sono poi stati infine cancellati dalla repository per ridurne l’occupazione di spazio.  
Nelle fasi finali del progetto sono state dunque condotte tre run di cui una per la Resnet che è stata addestrata sugli spettrogrammi e due per la Cnn “standard” che è stata addestrata su MFCC e spettrogrammi.

Si è mantenuto il dataset così come viene fornito, diviso come già detto in precedenza in 77.73% training, 15.58% validation e 6.68% testing, per un totale di circa 70 mila tracce audio. Tra queste si è deciso di selezionare quelle di dimensione maggiore di 34 kB per le run sugli spettrogrammi, mentre per la run sugli MFCC si è fatto zero padding sulle tracce audio di lunghezza minore di cinque secondi e scartato quelle di lunghezza maggiore.

I parametri fissati per ogni esperimento sono i seguenti:

* Batch size = 32
* max epoche = 200 (100 per run MFCC) con una patience di 10 epoche valutata sulla validation loss
* loss function = binary\_crossentropy
* dropout ratio = 0.1 dopo ogni layer convoluzionale e 0.3 dopo ogni layer fully connected
* learning rate = 0.0001
* numero layer convoluzionali = 3
* nei layer convolutivi, regolarizzazione l2 con lambda = 0.001
* 2 layer max pooling di kernel 3x3 e stride 2x2, un layer di dimensione 2x2 e stride 2x2
* 64 neuroni nel primo layer FC, 32 neuroni nel secondo, singolo neurone nell’output layer
* per gli MFCC: numero MFCC = 20, hop length = 512, numero fft = 2048
* per la STFT frame length = 255 e frame step 128
* output sequence length per gli audio caricati nel tf.Dataset = 50000

La ResNet usata in questo progetto è un’architettura fissa, simile all’implementazione nel paper originale [2], mentre per la Cnn viene eseguita una run di tuning di parametri quali:

* Numero neuroni (minimo 16, massimo 64, incremento di 16 per trial)
* Dimensione kernel convoluzionale (minimo 2x2, massimo 5x5)

Per le run usando come feature MFCC, una volta selezionate le tracce audio in base al criterio di lunghezza, viene creato un JSON contenente per ogni file una label 0-1 e la lista dei valori degli MFCC. Questo viene caricato in due array Numpy che formano la coppia input-output su cui addestrare. Alla shape dell’input viene aggiunta una ulteriore dimensione di valore 1, rappresentante il numero di canali, per raggiungere la corretta dimensionalità per la rete convoluzionale. Finito il tuning e l’addestramento, il modello viene valutato sui dati di test e vengono registrate le metriche di accuracy, loss, precision, recall e viene calcolato l’F1 score. Viene inoltre salvato il grafico delle curve di apprendimento di loss e accuracy.

Per le run sugli spettrogrammi, le tre cartelle del dataset vengono direttamente caricate in un Tensorflow Dataset tramite la funzione audio\_dataset\_from\_directory. Il dataset inferisce automaticamente le label binarie da assegnare ai file in base alla cartella di provenienza. Sul tf.dataset viene applicata una funzione di mapping per ricavare gli spettrogrammi a partire dalla traccia audio, questo avviene calcolando la Short-time Fourier Transform e calcolandone il valore assoluto, ottenendo uno spettrogramma. Anche qui viene aggiunta la dimensione dei canali per poter passare il dataset alla rete neurale. Per migliorare i tempi di computazione vengono applicate ai dataset delle funzioni fornite dalla Tensorflow Data API. In particolare si applica il prefetch dei dati, così che al passo N di training vengano già letti i dati per il passo N+1. Il numero di dati da prefetchare è determinato dinamicamente dall’opzione tf.data.AUTOTUNE. Si applica poi il caching del database per ridurre il numero di operazioni da compiere e un’operazione di shuffle per allocare un buffer che ospiti un certo numero di file random.

Ottenuti i dataset viene iniziato il processo di tuning, addestramento e valutazione del modello come descritto in precedenza.

# **Risultati**

Si sono registrati i seguenti risultati:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Modello | Feature | Accuracy | Precision | Recall | F1 |
| Cnn | MFCC | 95.15 | 97.91 | 91.43 | 94.56 |
| Cnn | Spettrogrammi | 56.82 | 55.75 | 95.99 | 70.53 |
| ResNet | Spettrogrammi | 88.58 | 88.61 | 90.41 | 89.50 |
| VGG16[6] | Spettrogrammi | 68 | 81 | 68 | 64.50 |
| VGG16[7] | MFCC | 93 |  |  |  |
| LSTM[7] | MFCC | 91 |  |  |  |
| Cnn[14] | Mel Spect. | 71.14 |  |  |  |
| TCN[8] | Mel Spect. | 92 |  |  |  |
| STN[8] | Mel Spect. | 80 |  |  |  |
| Cnn[14] | MFCC | 74.54 |  |  |  |
| ResNet[14] | Mel Spect | 83.55 |  |  |  |

Questi risultati, se confrontati con i paper citati in precedenza, sono molto incoraggianti: la Cnn addestrata sui MFCC ha ottenuto performance superiori di tutti gli altri modelli citati, mentre la ResNet ha mostrato performance in linea o leggermente inferiori ad altri modelli stato dell’arte.

La rete convolutiva “standard”, quando addestrata sugli spettrogrammi, non ottiene performance superlative, sebbene si sia mostrato in altri paper la difficoltà di queste reti ad apprendere la task

# **Conclusioni**

Con queste metriche si ritiene raggiunto l’obiettivo prefissato da questo progetto, con la consapevolezza che uno studio ancora più approfondito potrebbe portare ad ulteriori miglioramenti, sia dal lato delle performance che da quello dell’efficienza computazionale.

# **Riferimenti**

1. Bregler, C., Covell, M., Slaney, M.: Video Rewrite: Driving visual speech with audio. In: Proceedings of the ACM SIGGRAPH Conference on Computer Graphics (1997). https://doi.org/10.1145/3596711.3596787.

2. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (2016). https://doi.org/10.1109/CVPR.2016.90.

3. Masood, M., Nawaz, M., Malik, K.M., Javed, A., Irtaza, A., Malik, H.: Deepfakes generation and detection: state-of-the-art, open challenges, countermeasures, and way forward. Applied Intelligence. 53, (2023). https://doi.org/10.1007/s10489-022-03766-z.

4. Reimao, R., Tzerpos, V.: FoR: A dataset for synthetic speech detection. In: 2019 10th International Conference on Speech Technology and Human-Computer Dialogue, SpeD 2019 (2019). https://doi.org/10.1109/SPED.2019.8906599.

5. Wang, R., Juefei-Xu, F., Huang, Y., Guo, Q., Xie, X., Ma, L., Liu, Y.: DeepSonar: Towards Effective and Robust Detection of AI-Synthesized Fake Voices. In: MM 2020 - Proceedings of the 28th ACM International Conference on Multimedia (2020). https://doi.org/10.1145/3394171.3413716.

6. Papastergiopoulos, C., Vafeiadis, A., Papadimitriou, I., Votis, K., Tzovaras, D.: On the Generalizability of Two-dimensional Convolutional Neural Networks for Fake Speech Detection. In: MAD 2022 - Proceedings of the 1st International Workshop on Multimedia AI against Disinformation (2022). https://doi.org/10.1145/3512732.3533585.

7. Hamza, A., Javed, A.R., Iqbal, F., Kryvinska, N., Almadhor, A.S., Jalil, Z., Borghol, R.: Deepfake Audio Detection via MFCC features using Machine Learning. IEEE Access. (2022). https://doi.org/10.1109/ACCESS.2022.3231480.

8. Khochare, J., Joshi, C., Yenarkar, B., Suratkar, S., Kazi, F.: A Deep Learning Framework for Audio Deepfake Detection. Arab J Sci Eng. 47, (2022). https://doi.org/10.1007/s13369-021-06297-w.

9. Lea, C., Vidal, R., Reiter, A., Hager, G.D.: Temporal convolutional networks: A unified approach to action segmentation. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) (2016). https://doi.org/10.1007/978-3-319-49409-8\_7.

10. Jaderberg, M., Simonyan, K., Zisserman, A., Kavukcuoglu, K.: Spatial transformer networks. In: Advances in Neural Information Processing Systems (2015).

11. Oord, A. van den, Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., Kavukcuoglu, K.: WaveNet: A Generative Model for Raw Audio Based on PixelCNN Architecture. ArXiv. (2016).

12. Ping, W., Peng, K., Gibiansky, A., Arık, S., Kannan, A., Narang, S., Raiman, J., Miller, J.: Deep Voice 3: Scaling text-to-speech with convolutional sequence learning. In: 6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings (2018).

13. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) (2016). https://doi.org/10.1007/978-3-319-46493-0\_38.

14. Reimao, R., Tzerpos, V.: Synthetic speech detection using neural networks. In: 2021 11th International Conference on Speech Technology and Human-Computer Dialogue, SpeD 2021 (2021). https://doi.org/10.1109/SpeD53181.2021.9587406.