**Riconoscimento di volti generati da una GAN**

L. Naddei, R. Piccolo, M. Monti

## 1. Introduzione

L'avvento del Deep Learning ha comportato miglioramenti significativi nel campo della computer vision ed è stato applicato con successo per risolvere una vasta gamma di problemi complessi. In particolare, i rapidi avanzamenti nelle tecniche di sintesi di immagini "profonde" basate sul Deep Learning, come le Reti Generative Avversarie (GAN), hanno sollevato grande interesse e preoccupazione nell'opinione pubblica circa le sue implicazioni verso la società. Stiamo entrando in un'era in cui sarà impossibile distinguere ciò che è vero da ciò che è falso? Nella migliore delle ipotesi, questo potrebbe tradursi in una minaccia alla privacy, alla democrazia e alla sicurezza nazionale. Si tratta, pertanto, di uno dei temi più critici nella società digitale. Le GAN hanno introdotto miglioramenti apprezzabili in termini di qualità delle immagini. Esistono metodi basati su GAN sia per generare immagini da zero che per modificare caratteristiche di un'immagine già esistente. A tale scopo, sono state sviluppate un gran numero di applicazioni interessanti. Ad ogni modo, questa tecnologia può essere usata per scopi malevoli, per esempio per generare profili e notizie false sul web. Anche l'osservatore più attento può essere, difatti, tratto in inganno se posto di fronte ad immagini generate da GAN. Risulta, quindi, indispensabile la proposta di tecnologie che possano rilevare in maniera automatica e di seguito stabilire l'integrità di contenuto visivo digitale. Di recente, un gran numero di rilevatori di immagini "profonde" sono stati proposti in letteratura, ciascuno implementando uno dei metodi dello stato dell'arte. Ad ogni modo, tutti sembrano trasmettere l'idea che il compito assegnato non sia impegnativo. Invero, la pratica suggerisce che allo scomparire di condizioni favorevoli (assenza di distorsione nelle immagini, allineamento dei dati di training) le prestazioni calano in maniera drammatica e ciò è esperienza comune sul web, dove le immagini sono compresse e ricompresse, così distruggendo dettagli e tracce preziose.

## 2. Approccio implementato

Obiettivo di questo lavoro è realizzare un classificatore capace di discernere volti reali da volti sintetici e quindi valutarne la bontà in termini di generalizzazione se confrontato con immagini generate da altre reti. Le tecniche più performanti propongono soluzioni basate su Reti Neurali Convoluzionali (CNN).

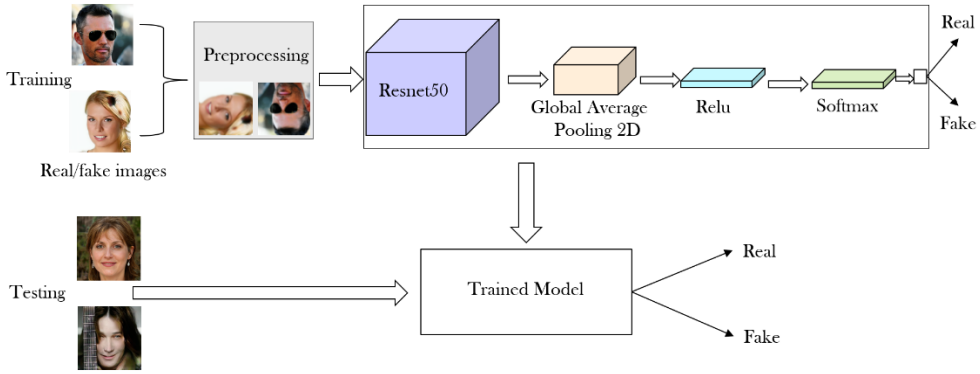
Ad oggi, i metodi dello stato dell'arte maggiormente impiegati sono tre:

- Approccio basato sull'apprendimento supervisionato di caratteristiche nel dominio spaziale;
- Approccio basato sull'apprendimento supervisionato di caratteristiche nel dominio frequenziale;
- Approccio basato sull'apprendimento di caratteristiche che "generalizzano".

I primi due, quelli interamente supervisionati, si rivelano particolarmente efficaci solo nel momento in cui le immagini GAN in fase di test provengo da un modello che è anche presente in fase di addestramento. Tuttavia, si scoprono fallimentari se chiamati a generalizzare su dati prodotti da modelli mai visti prima. Pertanto, nuovi metodi sono stati proposti nel tentativo di indirizzare questa problematica.

L'idea, proposta in [1] e alla quale si ispira il nostro progetto, prevede di realizzare augmentation tramite un'operazione di filtraggio gaussiano in fase di pre-processamento in modo da "forzare" il discriminatore ad apprendere caratteristiche quanto più generali. Tale azione si accompagna al training di un modello standard pre-addestrato su ImageNet, ResNet50, insieme con una strategia di fine-tuning. Il fine-tuning prevede il riuso delle caratteristiche di più basso livello (quelle comuni alla stragrande maggioranza delle immagini) in un diverso scenario, per poi procedere al riaddestramento degli ultimi strati della rete, quelli che estraggono feature specifiche all'applicazione in esame. Restringendo l'attenzione a quanto fatto nel nostro lavoro, l'idea è stata quella di "congelare" (si intende, rendere non addestrabili) i primi 35 strati del modello base e di non includerne gli ultimi fully-connected, di modo da poter adattare opportunamente la base al nostro problema specifico. In particolare, sono stati inclusi uno strato di GlobalAveragePooling2D (con lo

scopo di ridurre la complessità degli strati successivi effettuando un'operazione di media aritmetica sulle singole patch di input) e 1-2 strati Dense (strato fully-connected, dove tutti i nodi dello strato precedente presentano connessioni pesate verso quelli di uscita) con funzioni di attivazione, rispettivamente, Relu e Softmax.




## 3. Dataset

Nel nostro lavoro la rete è stata addestrata e testata su un dataset fornitoci da [2] e costituito da 4k immagini di volti reali e 10k di volti sintetici relativi a cinque differenti reti generative avversarie: RelGAN, StartGAN, StyleGAN, Style2GAN, ProGAN.

Le immagini reali sono state divise in due sottogruppi di cardinalità 2000. Ciascun sottogruppo abbraccia immagini di risoluzione, rispettivamente, 256x256 pixel e 1024x1024 piel. Di contro, le immagini RelGAN, StartGAN, ProGAN hanno risoluzione 256x256 pixel, quelle StyleGAN e Style2GAN di 1024x1024. Abbiamo suddiviso il dataset tramite il modulo Python "spiltfolders" in tre sottoinsiemi: training, validation, test. È stato considerato il 70% del totale per il training, il 10% per la validazione e il 20% per il test, avendo preventivamente escluso le immagini ProGAN dall'insieme di partenza.

Per preparare i dati è stata utilizzata la funzione di Keras "ImageDataGenerator" con il metodo flow_from_directory. Per tutti i set, le immagini sono state riportate nel range [0,1] e ritagliate al centro di 224 × 224 pixel. Solo per il set di training sono state previste le operazioni di data-augmentation quali la rotazione random di angoli multipli di 90° e il blurring gaussiano con sigma variabile nel range [0.0, 3.0].

In fase di test, sono state valutate le prestazioni in termini di Area Under ROC Curve (ovvero, misura della capacità di un classificatore di distinguere tra classi e viene utilizzata come riepilogo della ROC[1]) e di accuracy (ovvero, la percentuale delle istanze classificate correttamente) per ogni GAN. Sono, quindi, state incluse le immagini ProGAN precedentemente escluse per stabilire la capacità di generalizzazione della rete.

| Nome | Contenuto | # Immagini |
|---|---|---|
| RelGAN | Generated faces (CelebA) | 2000 |
| StartGAN | Generated faces (CelebA) | 2000 |
| StyleGAN | Generated faces (FFHQ) | 2000 |
| Style2GAN | Generated faces (FFHQ) | 2000 |
| ProGAN | Generated faces (CelebA-HQ) | 2000 |

[1] La ROC è utilizzata principalmente per la classificazione binaria. È una curva di probabilità che traccia il TPR (True Positive Rate) contro FPR (False Positive Rate) a vari valori di soglia e separa essenzialmente il "segnale" dal "rumore".

## 4. Risultati sperimentali

Una delle principali difficoltà da affrontare è stato l'overfitting, ovvero, il modello costruito si adattava troppo ai dati di training, senza, quindi, riuscire a generalizzare, neanche in fase di validazione. Inizialmente, il modello manifestava overfitting nell'arco di 3-4 epoche (ovvero, l'accuracy in validazione presentava un andamento a zig-zag). Per venire a capo del problema, intrinsecamente e principalmente legato al valore assegnato al "tasso di apprendimento", abbiamo adottato una strategia ispirata all'articolo [3], in virtù della quale si fissa un valore iniziale per il learning rate (il miglior risultato trovato è stato $3e^{-5}$ tra $e^{-4}$, $2e^{-5}$, $e^{-5}$), quindi si procede e lo si decrementa di epoca in epoca su di arco di 12 epoche di una quantità prefissata ($e^{-1/10}$). Tutto questo è stato accompagnato dalla scelta di Adam, nella sua variante AMSgrad, come "ottimizzatore". In termini di dimensione del batch, i valori testati sono stati {30,32,64,128,512} e tra questi quello che ha garantito risultati, tutto sommato, soddisfacenti è stato *batch_size = 64*, specie se in combinazione con un numero di epoche pari a 12 (tra 2,5,8,10 e 20) e la scelta di adoperare un unico strato Dense (vedi Figura 1), in luogo di due (vedi Figura 2).

```
Epoch 00001: LearningRateScheduler reducing learning rate to 2.9999999242136255e-05.
132/132 [==============================] - 358s 3s/step - loss: 0.2890 - accuracy: 0.8745 - auc: 0.9484 - val_loss: 0.6709 - val_accuracy: 0.7925 - val_auc: 0.8773
Epoch 2/12

Epoch 00002: LearningRateScheduler reducing learning rate to 2.9999999242136255e-05.
132/132 [==============================] - 344s 3s/step - loss: 0.2740 - accuracy: 0.8779 - auc: 0.9534 - val_loss: 0.6063 - val_accuracy: 0.7408 - val_auc: 0.8263
Epoch 3/12

Epoch 00003: LearningRateScheduler reducing learning rate to 2.714512185533531e-05.
132/132 [==============================] - 343s 3s/step - loss: 0.2582 - accuracy: 0.8845 - auc: 0.9590 - val_loss: 0.4779 - val_accuracy: 0.7775 - val_auc: 0.8664
Epoch 4/12

Epoch 00004: LearningRateScheduler reducing learning rate to 2.4561922698235563e-05.
132/132 [==============================] - 343s 3s/step - loss: 0.2365 - accuracy: 0.8963 - auc: 0.9656 - val_loss: 0.3877 - val_accuracy: 0.8392 - val_auc: 0.9212
Epoch 5/12

Epoch 00005: LearningRateScheduler reducing learning rate to 2.2224547039485775e-05.
132/132 [==============================] - 343s 3s/step - loss: 0.2393 - accuracy: 0.8936 - auc: 0.9647 - val_loss: 0.2862 - val_accuracy: 0.8883 - val_auc: 0.9550
Epoch 6/12

Epoch 00006: LearningRateScheduler reducing learning rate to 2.0109601849130354e-05.
132/132 [==============================] - 343s 3s/step - loss: 0.2178 - accuracy: 0.9037 - auc: 0.9710 - val_loss: 0.7608 - val_accuracy: 0.6867 - val_auc: 0.7719
Epoch 7/12

Epoch 00007: LearningRateScheduler reducing learning rate to 1.8195920985944437e-05.
132/132 [==============================] - 341s 3s/step - loss: 0.1989 - accuracy: 0.9126 - auc: 0.9758 - val_loss: 0.9318 - val_accuracy: 0.5908 - val_auc: 0.6899
Epoch 8/12

Epoch 00008: LearningRateScheduler reducing learning rate to 1.6464349585886427e-05.
132/132 [==============================] - 341s 3s/step - loss: 0.1964 - accuracy: 0.9156 - auc: 0.9765 - val_loss: 0.2237 - val_accuracy: 0.9000 - val_auc: 0.9690
Epoch 9/12

Epoch 00009: LearningRateScheduler reducing learning rate to 1.489755972245435e-05.
132/132 [==============================] - 342s 3s/step - loss: 0.1818 - accuracy: 0.9226 - auc: 0.9798 - val_loss: 0.2439 - val_accuracy: 0.8992 - val_auc: 0.9655
Epoch 10/12

Epoch 00010: LearningRateScheduler reducing learning rate to 1.3479869358821577e-05.
132/132 [==============================] - 339s 3s/step - loss: 0.1837 - accuracy: 0.9215 - auc: 0.9794 - val_loss: 0.4183 - val_accuracy: 0.8275 - val_auc: 0.9099
Epoch 11/12

Epoch 00011: LearningRateScheduler reducing learning rate to 1.2197090103041846e-05.
132/132 [==============================] - 339s 3s/step - loss: 0.1660 - accuracy: 0.9306 - auc: 0.9835 - val_loss: 0.6027 - val_accuracy: 0.8217 - val_auc: 0.8998
Epoch 12/12

Epoch 00012: LearningRateScheduler reducing learning rate to 1.1036383192702672e-05.
132/132 [==============================] - 340s 3s/step - loss: 0.1643 - accuracy: 0.9345 - auc: 0.9836 - val_loss: 0.2216 - val_accuracy: 0.9100 - val_auc: 0.9733
1e-05
```

Figura 1: batch_size=64, epochs=12, lr=3e-05, 1 strato Dense.

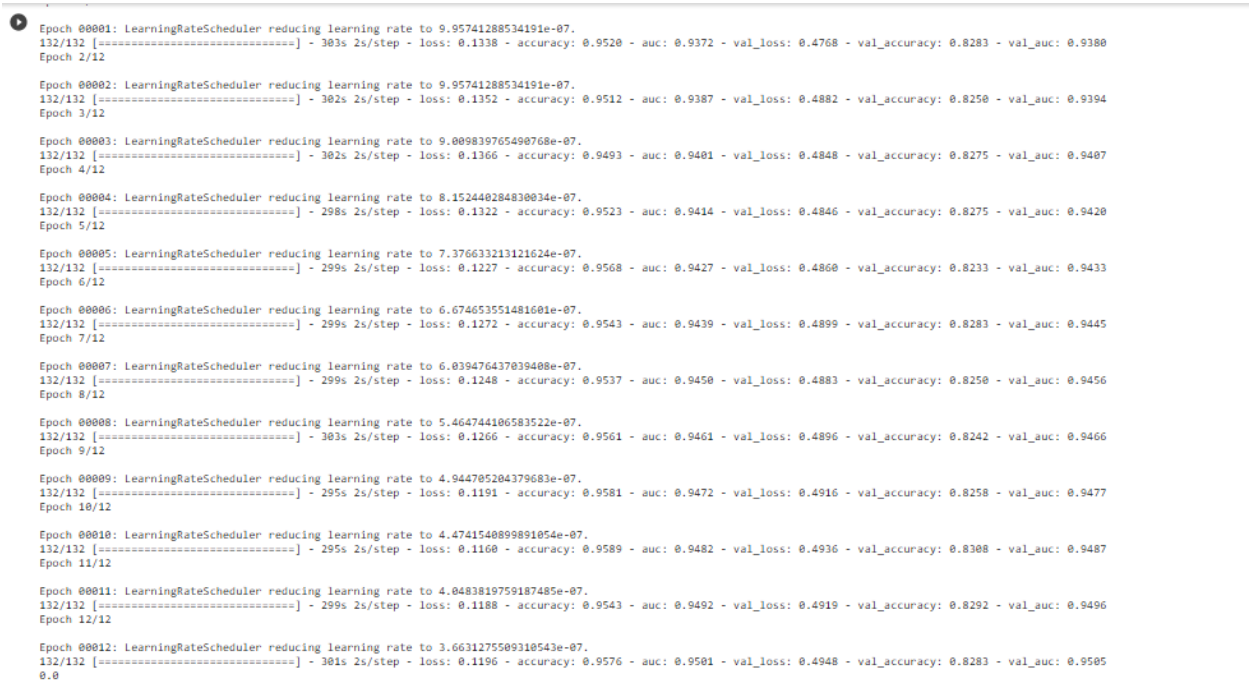



Figura 2: batch_size=64, epochs=12, lr=3e-05, 2 strati Dense.

Per la generalizzazione, riducendo il test-set alle sole immagini ProGAN, i risultati ottenuti sono riportati in Figura 3 e 4:


Figura 3: caso uno strato Dense.


Figura 4: caso due strati Dense.

Si nota come ci siamo discostati molto dai valori ottenuti sulle singole GAN, che mostriamo di seguito:

| | Accuracy | Area Under Curve (AUC) |
|---|---|---|
| StyleGAN | 0.8316 | 0.9100 |
| Style2GAN | 0.8883 | 0.9548 |
| StartGAN | 0.8874 | 0.9558 |
| RelGAN | 0.8799 | 0.9500 |

Ovvero, la rete mal sopporta il cambio di dominio.

Quello che abbiamo potuto riscontrare, tuttavia, è che è proprio quando l'accuracy supera l'80% che la capacità di generalizzazione della rete diminuisce drasticamente, come illustrato a seguire:

```
/usr/local/lib/python3.7/dist-packages/keras/engine/training.py:1915: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generat
  warnings.warn('`Model.fit_generator` is deprecated and '
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:13: DeprecationWarning: This function is deprecated. Please call randint(1, 4 + 1) instead
  del sys.path[0]
Epoch 1/8
132/132 [==============================] - 450s 3s/step - loss: 0.6294 - accuracy: 0.6590 - auc_3: 0.7165 - val_loss: 0.6846 - val_accuracy: 0.5725 - val_auc_3: 0.5850
Epoch 2/8
132/132 [==============================] - 417s 3s/step - loss: 0.5774 - accuracy: 0.6974 - auc_3: 0.7659 - val_loss: 0.6941 - val_accuracy: 0.5367 - val_auc_3: 0.5759
Epoch 3/8
132/132 [==============================] - 414s 3s/step - loss: 0.5611 - accuracy: 0.7021 - auc_3: 0.7812 - val_loss: 0.6183 - val_accuracy: 0.6850 - val_auc_3: 0.7580
Epoch 4/8
132/132 [==============================] - 411s 3s/step - loss: 0.5366 - accuracy: 0.7184 - auc_3: 0.8027 - val_loss: 0.5542 - val_accuracy: 0.7333 - val_auc_3: 0.8016
Epoch 5/8
132/132 [==============================] - 413s 3s/step - loss: 0.5303 - accuracy: 0.7277 - auc_3: 0.8079 - val_loss: 0.5257 - val_accuracy: 0.7392 - val_auc_3: 0.8206
Epoch 6/8
132/132 [==============================] - 416s 3s/step - loss: 0.4895 - accuracy: 0.7626 - auc_3: 0.8430 - val_loss: 0.5163 - val_accuracy: 0.7483 - val_auc_3: 0.8225
Epoch 7/8
132/132 [==============================] - 417s 3s/step - loss: 0.4584 - accuracy: 0.7815 - auc_3: 0.8638 - val_loss: 0.5071 - val_accuracy: 0.7467 - val_auc_3: 0.8302
Epoch 8/8
132/132 [==============================] - 418s 3s/step - loss: 0.4320 - accuracy: 0.7982 - auc_3: 0.8807 - val_loss: 0.5691 - val_accuracy: 0.7517 - val_auc_3: 0.8536
3e-05
```

```
44/44 [==============================] - 33s 741ms/step - loss: 0.6299 - accuracy: 0.7314 - auc_3: 0.7927

Test accuracy: 0.7314285635948181,
Area under curve: 0.7926857471466064
```

| | Accuracy | Area Under Curve (AUC) |
|---|---|---|
| StyleGAN | 0.7542 | 0.8470 |
| Style2GAN | 0.7600 | 0.8589 |
| StartGAN | 0.7508 | 0.8414 |
| RelGAN | 0.7850 | 0.9555 |

Come si può notare, la percentuale di accuracy non superava il 79% in training e il 75% in validazione, allorchè in fase di analisi di robustezza ottieniamo un 73% di accuracy, non molto lontano da quanto ottenuto in validation. La rete ha conservato le sue capacità classificatorie anche cambiando dominio, ovvero, anche quando messa a confronto con immagini prodotto di un'architettura cui non era stata esposta prima.

Quali le possibili soluzioni alternative? Sulla scia di [4], l'idea potrebbe essere di irrobustire la fase di pre-processamento, accompagnando il blurring gaussiano con una compressione JPEG. O, ancora, restringendoci alle sole immagini di persone esistenti ma manipolate (ergo, RelGAN, StartGAN ed eventualmente ProGAN), si potrebbe pensare di effettuare un cambiamento di prospettiva e chiedersi non più "è questa un'immagine vera o falsa?", bensì "è la persona soggetto di questa immagine effettivamente quella persona?", idea che si ispira al lavoro [5].

## Riferimenti bibliografici


[1] Xuan, B. Peng, W. Wang, and J. Dong, "On the generalization of GAN image forensics," in Chinese Conference on Biometric Recognition, 2019, pp. 134–141.

[2] http://www.grip.unina.it/download/corso/GANfacesDataset.zip

[3] J. Jordan, "Setting the learning rate of your neural network", Aug. 2020. www.jeremyjordan.me/nn-learning-rate

[4] S.-Y. Wang, O. Wang, R. Zhang, A. Owens, and A. Efros, "CNN-generated images are surprisingly easy to spot... for now," IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2020.

[5] Rössler, A., Cozzolino, D., Verdoliva, L., Riess, C., Thies, J., Nießner, M.: Faceforensics++: Learning to detect manipulated facial images. In: ICCV 2019 (2019).

[6] D. Gragnaniello, D. Cozzolino, F. Marra, G. Poggi, and L. Verdoliva, "Are GAN generated images easy to detect? A critical analysis of the state-of-the-art," IEEE International Conference on Multimedia & Expo (ICME), 2021.

[7] L. Verdoliva, "Media Forensics and DeepFakes: An Overview," in *IEEE Journal of Selected Topics in Signal Processing*, vol. 14, no. 5, pp. 910-932, Aug. 2020.