Unmasking Face

Mattia Bencivenga
Ingegneria informatica
276504

Ciro Maiello
Ingegneria informatica
276765

Ivan Orefice
Ingegneria informatica
276506

Abstract

In tempi di pandemia COVID-19 ci siamo ritrovati a dover sempre più far uso di mascherine sanitarie, le quali coprono parte del nostro viso rendendoci spesso poco riconoscibili sia da sistemi di videosorveglianza con riconoscimento facciale che da persone. Ciò che questo modello prova a fare è un task di Image Inpainting: rimuovere la mascherina dal viso e successivamente provare a ricreare la porzione di viso precedentemente coperta nel modo più credibile e coerente possibile.

1. Introduzione

Abbiamo elaborato un modello capace di segmentare le mascherine presenti all'interno delle immagini. Tale sistema, poi, provvede alla cancellazione e successiva generazione della porzione di viso al di sotto della mascherina, a partire da un'immagine di un viso con mascherina e relativa mappa di segmentazione (*Figura 1*).
Il modello sviluppato è diviso in due moduli principali: il *Map Module*, una versione modificata della U-Net [1], che si occupa della segmentazione, e l'*Editing Module*, che si occupa della ricostruzione del viso. Quest'ultimo comprende una Generative Adversarial Network (GAN) [2], che usa come parte generativa una U-Net mentre si ispira alla Pix2Pix [3] per la parte discriminativa (in particolare utilizziamo due discriminatori), ed una Perceptual Network.

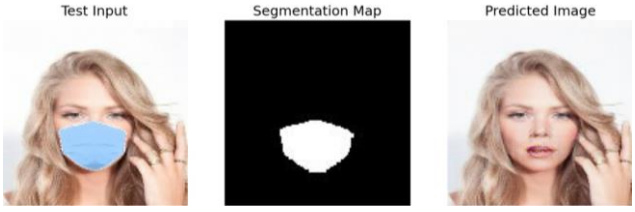

Figura 1: Esempio di rimozione di mascherina in fase di test.

1.1. Lavori correlati

Nel panorama del Machine Learning e della Computer Vision è presente una moltitudine di lavori che si pongono come obiettivo la ricostruzione di porzioni di immagini. Nel corso della nostra ricerca ci siamo imbattuti nel lavoro "A Novel GAN-Based Network for Unmasking of Masked Face" [4] a cui ci siamo ispirati e da cui abbiamo studiato sia le architetture utilizzate (U-Net e Pix2Pix) che le diverse metriche di valutazione adottate.

2. Dataset

Poiché questo genere di task sta acquisendo particolare importanza solo nell'ultimo periodo, in rete non sono disponibili dataset che offrono coppie di immagini di volti con e senza mascherina; abbiamo dovuto quindi creare, a partire da un dataset di volti di celebrità di dimensione fissa 178x218 (CelebA [5]), la controparte del viso con la mascherina. Il dataset di partenza contiene immagini a colori che sono poi state ridimensionate a 128x128 (per ragioni di performance); il dataset sintetico da noi utilizzato è stato ottenuto tramite un tool automatico chiamato MaskTheFace [6]: un sistema capace di aggiungere in modo coerente vari tipi di mascherine a volti. Abbiamo scelto tre tipi di mascherine fra le più utilizzate: la mascherina chirurgica, quella di stoffa e la KN95. La divisione del dataset è stata fatta in proporzione alla diffusione della mascherina nella realtà; quindi, quella chirurgica comprende il 50% del dataset mentre quella di stoffa e la KN95 comprendono il 25% del dataset ciascuna. Essendo il tool utilizzato soggetto ad errori di vario tipo, si è dovuto effettuare un filtraggio per scartare le mascherine che non erano state inserite in maniera corretta, come è possibile notare in *Figura 2*.

È poi stata necessaria la creazione delle mappe di segmentazione delle mascherine: esse sono state ottenute tramite uno script creato da noi che effettua il valore assoluto della differenza tra la singola immagine con mascherina e la sua controparte senza. Abbiamo infine rimosso il rumore da tali mappe tramite operazioni di *Morphological Processing*: a questo scopo abbiamo utilizzato la *opening*, ovvero un meccanismo di *erosion* seguito da una *dilation*. Pertanto, il nostro dataset è composto da tre parti: le immagini con mascherina, le relative segmentazioni per il Map Module e le immagini originali da CelebA per l'Editing Module. Abbiamo diviso il dataset per la segmentazione in 80% per il training ed il restante 20% per la validation.

Figura 2: Esempi di errori di MaskTheFace.

Il dataset iniziale era composto da 200mila immagini per parte (quindi un totale di 600mila immagini), ma per motivi computazionali la sua dimensione è stata diminuita a 20mila immagini per parte.

3. Metodo

Come già accennato nelle precedenti sezioni, l'architettura utilizzata è una riproduzione, con le dovute modifiche, al lavoro di N. U. Din *et al.* [4] ed è divisa in due moduli principali.

3.1. Map Module

Dopo la creazione del dataset, ci siamo concentrati sulla segmentazione delle mascherine. Il modello di riferimento utilizzava una U-Net modificata; in particolare, aggiunge all'architettura originale (formata da 9 blocchi) un blocco all'encoder e, parallelamente, uno al decoder, collegati da una skip connection, generando complessivamente un'architettura da 11 blocchi. A differenza degli autori della ricerca, abbiamo ritenuto opportuno utilizzare l'architettura base della U-Net, e non quella modificata, con un input size ed un output size di 128x128 su un canale. Questa scelta è stata influenzata sia da semplici ragioni di performance di Google Colaboratory [7] e del calcolatore in locale, sia dai risultati ottenuti, che vedremo nella sezione 4, i quali hanno mostrato valori di accuracy più che accettabili dopo poche epoche. Ciò è stato possibile grazie alla semplicità del task e all'omogeneità del dataset.

Figura 3: Risultati del Map Module in fase di test.

Nella nostra architettura, così come nella U-Net tradizionale, abbiamo che ogni convoluzione interna è seguita da una funzione di attivazione Leaky ReLU per prevenire problemi di vanishing gradient. In particolare, evita le difficoltà delle funzioni tanh e sigmoide, ossia per valori molto piccoli e molto grandi il processo di ottimizzazione è molto lento, e della ReLU (il dying ReLU). Inoltre, sono presenti: un livello di Batch Normalization che permette, attraverso tecniche di normalizzazione di *re-centering* e *re-scaling*, di velocizzare e stabilizzare la fase di training; un livello di Max Pooling che effettua downsampling; e infine, un livello di Dropout nei blocchi dell'encoder in modo da evitare il fenomeno di overfitting. Il primo blocco dell'encoder, invece, presenta una ReLU piuttosto che una Leaky ReLU come funzione di attivazione e non ha bisogno di Batch Normalization, essendo la ReLU già performante di suo. L'ultimo blocco è caratterizzato da funzioni di attivazione come tanh e sigmoide, le quali sono più adatte alla segmentazione di un singolo oggetto all'interno di un'immagine (problema di classificazione binaria); ciò, inoltre, ci ha spinto ad utilizzare una funzione di costo di tipo binary crossentropy.

3.2. Editing Module

Questo modulo è stato, fra i due, quello più impegnativo per noi sia per il design dell'architettura che per la sua effettiva implementazione. L'architettura è formata da una GAN che presenta una parte generativa simile a quella usata nel Map Module (un autoencoder ispirato alla U-Net) ed una discriminativa composta da due discriminatori: nello specifico, uno relativo a tutta l'immagine (*Whole Region Discriminator*) e un altro che invece si concentra in modo più dettagliato solo sulla regione coperta dalla mascherina (*Mask Region Discriminator*). Infine, all'interno di questo modulo, troviamo una VGG19 [8] utilizzata come Perceptual Network, che permette al generatore di creare features che siano più simili a quelle del ground truth.

3.2.1 Generatore

Inizialmente abbiamo cercato di riprodurre l'architettura della parte generativa seguendo il lavoro di N. U. Din *et al.* [4] con le modifiche precedentemente spiegate nel paragrafo 3.1, ossia con la riduzione da 11 blocchi convolutivi a 9. A differenza della U-Net implementata nel Map Module, in input non avremo più solo l'immagine mascherata (128x128 ad un canale), ma la stessa (128x128 a tre canali) concatenata con la mappa di segmentazione relativa generata dal Map Module. In output, invece, avremo un'immagine 128x128 a tre canali. Inoltre, troviamo l'introduzione di un blocco definito *Squeeze and Excitation* [9] usato dopo il secondo blocco. Quest'unità è formata da un livello di Average Pooling e due livelli fully connected (attivati uno da ReLU e l'altro da sigmoide), e permette di aumentare la potenza di rappresentazione di una rete consentendole di eseguire una ricalibrazione dinamica delle feature a livello dei canali. Un'altra differenza rispetto al modulo precedente è che il blocco di bottleneck è formato da quattro livelli di *atrous convolution* [10]: una

convoluzione che permette di rendere la generazione di parti mancanti più coerente con il resto dell'immagine, catturando di volta in volta porzioni più grandi della stessa. Purtroppo, questo tipo di architettura è risultata troppo pesante per le capacità computazionali della GPU sulla quale è stato allenato il modello (ne discuteremo nella sezione 4); di conseguenza, abbiamo dovuto alleggerire la struttura passando da 9 a 5 blocchi. In particolare, abbiamo lasciato: i primi due livelli, quello centrale con le atrous convolution, e gli ultimi due connessi ai primi tramite skip connection; abbiamo poi dovuto eliminare il blocco di Squeeze and Excitation.

3.2.2 Discriminatori

I discriminatori si basano sull'architettura discriminativa della Pix2Pix. Presentano, quindi, 5 livelli convolutivi seguiti da Batch Normalization e Leaky ReLU, tranne che per il primo livello, a cui manca la Batch Normalization, e l'ultimo livello, a cui manca sia la Batch Normalization che la Leaky ReLU. L'unica differenza tra i due discriminatori riguarda gli input. In particolare, il Whole Region Discriminator prende in ingresso l'immagine creata dal generatore o l'immagine di ground truth (128x128 a tre canali), mentre il Mask Region Discriminator prende in input:

$$I_{mask\_region} = I_{input} \otimes \left(1 - I_{mask\_map}\right) + \left(I_{edit} \otimes I_{mask\_map}\right)$$

dove I_{input} è l'immagine con la mascherina in ingresso al sistema, I_{mask_map} è la mappa di segmentazione ottenuta dal Map Module, e I_{edit} può essere o l'immagine generata o quella di ground truth.

3.2.3 Perceptual Network

L'ultima parte dell'Editing Module è la Perceptual Network. Questa rete viene usata per aiutare la generazione di immagini con feature quanto più simili a quelle di ground truth. Ciò è ottenuto con una VGG19 pre-allenata su ImageNet [11]: nella fattispecie abbiamo sfruttato le feature map di 3 livelli convolutivi intermedi (*block3_conv4*, *block4_conv4*, *block5_conv4*). In input riceve le immagini 128x128 a tre canali appena generate e le immagini di ground truth.

3.2.4 Loss

Allenare al meglio questo modulo significa far sì che la rete apprenda al meglio feature di alto livello di volti, come bocca, naso, mento e zigomi, così da ricreare al meglio la regione al di sotto della mascherina. Per far ciò, oltre alla loss di minmax generica della GAN, sono state usate altre loss: infatti, abbiamo quella della Perceptual Network e una custom loss, chiamata *reconstruction loss*, così definita:

$$\mathcal{L}_{rc} = \mathcal{L}_{SSIM} + \mathcal{L}_{l1}$$

La loss del generatore è una non-saturating loss [12]:

$$\mathcal{L}_{adv} = \mathbb{E}_{I_{edit} \in S}\left[\log\left(D\left(G\left(I_{input}, I_{mask\_map}\right)\right)\right)\right]$$

Tale loss punta a massimizzare le probabilità che la generazione di immagini venga etichettata come vera dal discriminatore piuttosto che minimizzare la probabilità che il discriminatore etichetti come falsa un'immagine generata. Ciò viene raggiunto massimizzando questa loss implementata sia per la parte adversarial del Whole Region Discriminator che per quella del Mask Region Discriminator.

La loss della Perceptual Network si basa, come già detto, sulle feature map dei livelli intermedi della VGG19: cerca quindi di ottenere informazioni d'alto livello del volto cercando di generare visi plausibili.

Queste loss vengono poi combinate linearmente per permettere alla parte generativa del modello di apprendere al meglio come generare la parte di volto coperta:

$$\mathcal{L}_{comp} = \lambda_{rc}\left(\mathcal{L}_{rc} + \mathcal{L}_{perc}\right) + \lambda_{whole}\left(\mathcal{L}_{D}^{whole} + \mathcal{L}_{adv}^{whole}\right) + \lambda_{mask}\left(\mathcal{L}_{D}^{mask} + \mathcal{L}_{adv}^{mask}\right)$$

dove gli iperparametri λ_{rc} , λ_{whole} e λ_{mask} valgono rispettivamente 100, 0.3 e 0.7. Tali valori sono gli stessi utilizzati dagli autori del paper originale ottenuti empiricamente.

Le loss di entrambi i discriminatori sono di tipo minmax:

$$\mathcal{L}_{D}^{whole} = \mathbb{E}_{I_{gt} \in O}\left[\log\left(D_{whole}\left(I_{edit}, I_{gt}\right)\right)\right] + \mathbb{E}_{I_{edit} \in S}\left[\log\left(1 - D_{whole}\left(G\left(I_{input}, I_{mask\_map}\right)\right)\right)\right]$$

$$\mathcal{L}_{D}^{mask} = \mathbb{E}_{I_{gt} \in O}\left[\log\left(D_{mask}\left(I_{mask\_region}, I_{gt}\right)\right)\right] + \mathbb{E}_{I_{edit} \in S}\left[\log\left(1 - D_{mask}\left(G\left(I_{input}, I_{mask\_map}\right)\right)\right)\right]$$

Con I_{gt} immagine di ground truth.

4. Esperimenti

L'allenamento è stato effettuato separatamente per i due moduli implementati: spiegheremo prima l'allenamento del Map Module e successivamente quello dell'Editing Module, il quale fra i due è risultato il più difficoltoso, data anche la maggior complessità del task associato. Per quanto riguarda la prima parte dell'allenamento non abbiamo avuto particolari difficoltà: è stato sufficiente allenare la rete per 20 epoche in modo tale da ottenere risultati piuttosto soddisfacenti. Questo è stato possibile determinarlo non solo dai risultati visivi più che accettabili ma anche dai

valori della accuracy e della loss del modello, come possiamo vedere in *Figura 4*.

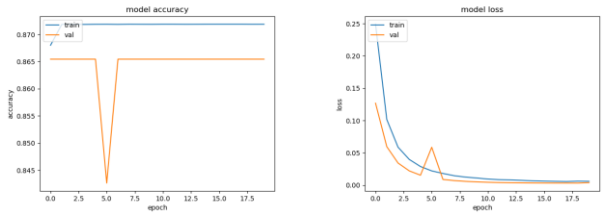

Figura 4: Andamento di accuracy e loss del Map Module in fase di training.

Questi risultati, come detto precedentemente, sono dovuti sia alla semplicità del task ma anche alla composizione del dataset che vede mascherine posizionate in posizioni estremamente simili rendendo ancora più semplice il lavoro della rete. Le immagini generate dalla rete sono state poi filtrate, così come descritto precedentemente per le mappe create da noi per la composizione del dataset (sezione 2).
Per quanto riguarda la seconda parte di allenamento dedicata all'Editing Module, inizialmente abbiamo ridotto la U-Net ad una rete a tre blocchi in quanto era troppo pesante computazionalmente per il calcolatore, con un ultimo layer convolutivo attivato da una sigmoide. Le prestazioni, però, non erano visivamente soddisfacenti: le immagini in uscita risultavano sbiadite con colori appiattiti, come si può notare in *Figura 5*.

Figura 5: Esempio di generazione con sigmoide in uscita.

Siamo riusciti a risolvere questo problema andando a modificare la funzione di attivazione in uscita poiché ci siamo resi conto che la normalizzazione applicata alle immagini in ingresso al modello non era coerente con la sigmoide che utilizzavamo come funzione di attivazione (valori compresi fra 0 ed 1). Quindi, tramite la sostituzione con una tanh (valori compresi fra -1 ed 1, coerenti con la normalizzazione applicate alle immagini in ingresso), siamo riusciti ad ottenere risultati decisamente migliori, come è possibile vedere in *Figura 6*.

Figura 6: Esempio di generazione con tanh in uscita.

Successivamente abbiamo aumentato il numero di blocchi da 3 a 5, ma questo risultava fatale per il calcolatore: l'allenamento incorreva in un Out of Memory. Dopo numerose prove, siamo riusciti a identificare il problema nel blocco di Squeeze and Excitation: eliminandolo, siamo riusciti a far lavorare il modello. In seguito, ci siamo soffermati sul cercare di capire quale tipo di loss utilizzare per il generatore: mentre il paper descriveva una loss negativa, noi abbiamo deciso di usare una loss positiva più coerente con quella vista nel paper della Pix2Pix a cui si ispira il nostro modello (Non-Saturating GAN Loss).

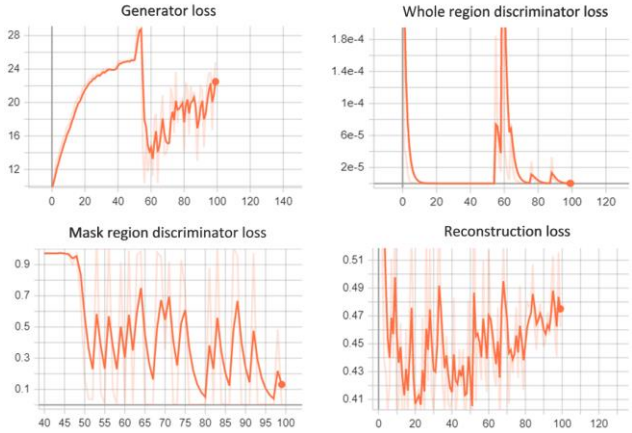

Figura 7: Andamento delle loss dell'Editing Module in fase di training.

Questo ha portato benefici non solo dal punto di vista visivo dei risultati ottenuti ad ogni epoca (le labbra generate erano decisamente migliori) ma anche dal punto di vista strettamente matematico, dandoci valori più conformi durante l'allenamento.
La parte discriminativa, come già detto in precedenza, si basa su quella della Pix2Pix ed è divisa in due discriminatori: il primo (Whole Region Discriminator) fornisce in uscita delle mappe di grandezza 14x14 e permette al generatore di generare porzioni di viso al di sotto della mascherina che siano al meglio integrate con il resto del viso; il secondo (Mask Region Discriminator), il quale viene introdotto dopo ⅔ dell'allenamento, lavora specificatamente solo sulla regione della mascherina e quindi dà un feedback più preciso su quella porzione di viso.
Questo modello è stato allenato per 100 epoche, delle quali le prime 40 sono durate 600 secondi ciascuna e le ultime 60 intorno agli 800 secondi ciascuna, fornendo delle immagini generate che, seppur non realistiche, sono assolutamente sensate. La rete è stata allenata usando una GPU Nvidia 1660Ti. Gli ottimizzatori utilizzati in tutta l'architettura sono di tipo Adam, con un learning rate di 0.001 per il Map Module e di 0.0002 per l'Editing Module.

Figura 8: Generazioni da immagini non appartenenti a CelebA.

5. Risultati e conclusioni

I risultati ottenuti usando un testset proveniente dal dataset CelebA con mascherine sintetiche possono essere considerati più che soddisfacenti. Questo è intuibile valutando non solo le immagini generate visivamente ma anche tramite metriche opportunamente implementate quali PSNR (Peak Signal-to-Noise Ratio) e SSIM (Structural Similarity Index Measure) [13]. Abbiamo scelto di usare questi due criteri di valutazione oggettivi poiché sono gli stessi utilizzati dagli autori del paper originale e anche quelli che abbiamo incontrato maggiormente durante il nostro corso di studi; inoltre, riteniamo che l'utilizzo combinato di queste due metriche possa dare una valutazione relativamente precisa sulla fedeltà delle immagini generate a partire da quelle di ground truth. I valori mostrati in *Tabella 1* sono la media dei valori delle metriche ottenuti durante la fase di test:

METODO	SSIM	PSNR
NOSTRO	0.621	22.71dB
PAPER ORIGINALE	0.864	26.19dB

Tabella 1: Comparazione delle performance dei metodi basata su SSIM e PSNR

Per quanto riguarda la SSIM, essa ha di base valori che variano da 0 ad 1 e nei casi migliori riusciamo ad ottenere valori maggiori di 0.8; per quanto concerne il PSNR, i valori tipici sono solitamente compresi da 20dB a 40dB, e noi riusciamo ad ottenere valori al più intorno ai 26dB.

Queste metriche non sono state calcolate però a partire da immagini provenienti dal mondo reale, poiché, da come si può notare in *Figura 8* e *Figura 9*, il nostro modello ha imparato a svolgere il proprio task sia su mascherine molto simili sia su volti posizionati allo stesso modo all'interno delle immagini. Si può notare, ad esempio, come la posizione della bocca sia sempre la stessa tra le diverse immagini del dataset. Quindi, il nostro modello non riesce a generalizzare al di fuori del dataset in maniera opportuna poiché affetto da overfitting. Questo problema si sarebbe potuto risolvere diminuendo il numero dei parametri oppure aumentando il dataset rendendolo più eterogeneo. Abbiamo cercato di intervenire sull'architettura, modificando il numero dei parametri e la complessità della rete, ma non abbiamo ottenuto i risultati sperati, anzi sono peggiorati. Una reale miglioria, in termini di generalizzazione, quindi, si sarebbe potuta ottenere andando a rendere il dataset molto più eterogeneo. Ciò, ovviamente, non è stato possibile poiché, essendo il dataset creato da noi stessi, sarebbe stato impraticabile in termini di tempo.

Figura 9: Generazioni da immagini appartenenti a CelebA con mascherine aggiunte sinteticamente.

Eventuali sviluppi futuri si basano sulle considerazioni appena fatte: ad esempio, con la creazione di un dataset ad hoc, tutto il modello, dalla segmentazione alla generazione, ne potrebbe giovare. Inoltre, con un hardware più prestazionale, si potrebbe rendere il modello più performante aggiungendo i blocchi eliminati dal modello originale (ad esempio, lo Squeeze and Excitation) per problemi computazionali.

6. Riferimenti

[1] O. Ronnerberger, P. Fischer e T. Brox, «U-Net: Convolutional Networks for Biomedical Image Segmentation,» 2015.

[2] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville e Y. Bengio, «Generative Adversarial Nets,» 2014.

[3] P. Isola, J.-Y. Zhu, T. Zhou e A. A. Efros, «Image-to-Image Translation with Conditional Adversarial Networks,» 2016.

[4] N. U. Din, K. Javed, S. Bae e J. Yi, «A Novel GAN-Based Network for Unmasking of Masked Face,» *IEEE Access*, vol. 8, 2020.

[5] Z. Liu, P. Luo, X. Wang e X. Tang, «Deep Learning Face Attributes in the Wild,» in *Proceedings of International Conference on Computer Vision (ICCV)*, 2015.

[6] A. Aqeel e R. Arijit, «Masked Face Recognition for Secure Authentication,» 2020.

[7] «Google Colaboratory,» Google, [Online]. Available: https://colab.research.google.com/.

[8] K. Simonyan e A. Zisserman, «Very Deep Convolutional Networks for Large-Scale Image Recognition,» 2014.

[9] J. Hu, L. Shen, S. Albanie, G. Sun e E. Wu, «Squeeze-and-Excitation Networks,» 2017.

[10] L.-C. Chen, G. Papandreou, F. Schroff e H. Adam, «Rethinking Atrous Convolution for Semantic Image Segmentation,» 2017.

[11] A. Krizhevsky, I. Sutskever e G. Hinton, «ImageNet Classification with Deep Convolutional,» 2012.

[12] M. Lucic, K. Kurach, M. Michalski, S. Gelly e O. Bousquet, «Are GANs Created Equal? A Large-Scale Study,» 2017.

[13] Z. Wang, A. C. Bovik, H. R. Sheikh e E. P. Simoncelli, «Image Quality Assessment: From Error Visibility to Structural Similarity,» in *IEEE Transactions Image Processing*, 2004, pp. 600-612.