

Attacchi avversari nelle reti neurali convoluzionali

Progetto ESM - Gruppo 06

Francesco Iannaccone, Matteo Conti

(a.a. 2020-2021)

Contents

1	Introduzione	1
1.1	Il problema	1
1.2	Difficoltà affrontate	2
2	Approccio implementato	2
2.1	Reti neurali utilizzate	2
2.2	Fast Gradient Signed Method	2
3	Dataset	3
3.1	Classificatore	4
4	Risultati sperimentali	5
4.1	Attacco non target	5
4.2	Attacco non target iterativo	5
4.3	Attacco target iterativo	6
4.4	Attacco black box	8

1 Introduzione

1.1 Il problema

Il nostro gruppo ha lavorato alla produzione di **esempi avversari**, ovvero alla creazione di impercettibili perturbazioni che, sovrapposte ad un' immagine in ingresso ad una rete neurale convoluzionale, siano in grado di causare un errore nella sua classificazione. Ad oggi, quello degli attacchi avversari è uno degli hot-topic per quanto riguarda i problemi di sicurezza nel *Deep Learning* e costantemente vengono proposte nuove tecniche di attacco e di difesa. L'obiettivo sarebbe quello di riuscire a realizzare reti neurali in grado di essere robuste a qualsiasi scenario possibile, ma al momento anche le reti

allo stato dell'arte, se non protette, sono vulnerabili a questi attacchi che ne minano la sicurezza e l'impiegabilità in campi safty-critical.

1.2 Difficoltà affrontate

Prima di poter però effettivamente testare l'efficacia degli attacchi avversari, è stato necessario allenare e testare una rete neurale che producesse in fase di test risultati soddisfacenti. Abbiamo dunque inizialmente organizzato correttamente il dataset, definito il modello della rete neurale e messo a punto sperimentalmente gli iperparametri, quali il learning-rate, il numero di epoche e il batch-size, per ottenere un'accuracy superiore al 90%.

Durante la fase di attacco, invece, abbiamo dovuto gestire il trade-off esistente tra ampiezza della perturbazione ed efficacia dell'attacco. In particolare, uno dei nostri obiettivi è stato quello di creare dei rumori visibili il meno possibile ad occhio nudo e che, allo stesso tempo, generassero un errore nella classificazione almeno 8 volte su 10.

2 Approccio implementato

2.1 Reti neurali utilizzate

Al fine di migliorare le prestazioni della rete neurale sui dati di test, abbiamo utilizzato la tecnica del **fine-tuning**: come rete convolutiva di base abbiamo utilizzato *ResNet152V2* pre-allenata su ImageNet a cui abbiamo aggiunto alcuni strati **Fully-Connected** che eseguono la classificazione delle immagini in ingresso.

La rete neurale è stata compilata con la loss function *CategoricalCrossentropy* e l'ottimizzatore *Adam*. Per quanto riguarda il **learning rate** della loss function, è stato selezionato il valore di default.

Per poter testare, come verrà descritto successivamente, attacchi di tipo **black-box**, abbiamo costruito una seconda rete neurale, sfruttando attraverso il fine-tuning la rete *VGG19*, addrestata su ImageNet. Dopo il modello base, sono stati aggiunti tre layer **Fully-Connected** intervallati da altri tre layer di BatchNormalization.

2.2 Fast Gradient Signed Method

L'approccio utilizzato per l'implementazione dell'attacco segue il metodo del FGSM presentato in [1], che si basa sul costruire l'attacco avversario a partire dal gradiente della *loss function* $J(\theta, y, x)$ rispetto all'ingresso x con la seguente formula:

$$\eta = \epsilon sign(\nabla_x J(\theta, y, x))$$

Lo scopo di tale perturbazione è di **massimizzare** la loss function e indurre la rete a compiere un errore di classificazione, alterando allo stesso tempo

in maniera **minima** l’aspetto dell’immagine in ingresso, garantendo infatti che $\|\eta\|_\infty \leq \epsilon$. Uno dei punti di forza di questa tecnica è che, utilizzando gli stessi framework utilizzati per l’allenamento delle reti, quali *keras*, *tensorflow* e molti altri, è possibile implementare in maniera semplice e efficiente questa tipologia di attacco.

Con una semplice modifica a questa tecnica, è possibile inoltre realizzare **attacchi target**, in cui le immagini sono perturbate al fine di essere classificate erroneamente come appartenenti ad una specifica classe scelta al momento dell’attacco. In questo caso la perturbazione η è calcolata secondo la seguente formula:

$$\eta = -\epsilon sign(\nabla_x J(\theta, y_t, x))$$

dove y_t rappresenta l’array di label target. Quindi, invece di costruire una perturbazione che aumenti la loss $J(\theta, y, x)$, si crea una perturbazione che minimizzi la $J(\theta, y_t, x)$ per la classe desiderata. Migliori performance si ottengono utilizzando una versione iterativa dell’attacco in cui vengono applicate più perturbazioni in sequenza ed eseguendo tra una iterazione e la successiva un’operazione di *clipping* con lo scopo di mantenere la perturbazione contenuta.

In conclusione, abbiamo realizzato un semplice attacco **black box** basato sul FGSM, costruendo l’attacco a partire da un modello surrogato. In questo caso, si suppone di non essere a conoscenza della struttura interna della rete e di conseguenza di non poter calcolare $\nabla_x J(\theta, y, x)$. Per aggirare questo problema, si può ipotizzare di realizzare l’attacco per una *rete surrogata*, allenata a partire dallo stesso dataset, della quale si conoscono la struttura interna e i parametri, ottenendo spesso un attacco in grado di compromettere il funzionamento anche della rete black box. Nella costruzione degli esempi avversari, l’attacco è generato sulla rete surrogata, ma le prestazioni vengono valutate sulla rete black box target.

3 Dataset

Inizialmente è stato utilizzato come dataset *LHI-Animal-Faces* descritto in [2], un dataset composto da circa 5000 immagini suddivise in 19 categorie di animali, volti umani e paesaggi naturali. Dopo aver suddiviso il dataset considerando 4/6 di immagini per il *training*, 1/6 per la *validation* e 1/6 per il *test*, mediante un apposito script python, e aver allenato la rete, a causa del ridotto numero di immagini e del grande numero di classi, le performance della rete sono risultate insoddisfacenti, non superando il 75% di accuracy: abbiamo dunque ritenuto opportuno utilizzare un dataset che contenesse più immagini al fine di ottenere prestazioni in fase di test migliori e di conseguenza poter testare gli attacchi su una rete più affidabile. E’ stato dunque impiegato il dataset *Animals-10*, contenente più di 30000 immagini con 10 classi di animali. Su questo dataset, entrambe le reti neurali allenate

hanno ottenuto performance soddisfacenti, ottenendo un'accuracy in fase di test prossima al 95%.



Figura 1: Alcuni esempi del dataset Animals-10

3.1 Classificatore

Attraverso l'uso di *ImageDataGenerator*, abbiamo caricato in maniera automatica le immagini di addestramento, validazione e test, effettuando la tecnica della **data augmentation** sulle immagini di training. In questo modo, attraverso il metodo *flow_from_directory*, abbiamo istanziato dei generatori che ci hanno permesso di caricare in maniera automatica le immagini, evitando di sovraccaricare la memoria.

Per quanto riguarda il training, abbiamo usato il metodo *fit_generator*, sfruttando i generatori definiti durante la fase di caricamento del dataset. Al fine di ottenere il miglior rendimento possibile dalla rete, è stato scelto un numero di **epoch** pari a 50.

Infine, abbiamo effettuato il test della rete neurale su un *batch* di 100 immagini. Dopo aver calcolato l'array di label predetto della rete e quello reale, sono state valutate le prestazioni della CNN in termini di accuratezza. In particolare, è stata elaborata la **matrice di confusione**, una matrice 10x10 che permette di capire quante immagini di test sono state classificate bene e quante invece sono state assegnate alla classe sbagliata. Testando la rete sull'intero test set di oltre 4000 immagini, si ottiene una *accuracy* del circa 96% e la seguente confusion matrix:

$$C = \begin{pmatrix} 709 & 4 & 2 & 5 & 0 & 7 & 14 & 3 & 4 & 1 \\ 1 & 388 & 0 & 0 & 0 & 0 & 8 & 2 & 0 & 0 \\ 0 & 1 & 215 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 315 & 0 & 0 & 0 & 0 & 6 & 0 \\ 5 & 3 & 0 & 4 & 453 & 0 & 9 & 0 & 2 & 0 \\ 9 & 2 & 3 & 0 & 0 & 230 & 5 & 1 & 1 & 8 \\ 1 & 0 & 1 & 0 & 0 & 0 & 272 & 6 & 0 & 0 \\ 5 & 1 & 1 & 0 & 0 & 2 & 20 & 244 & 0 & 0 \\ 1 & 3 & 0 & 2 & 1 & 0 & 1 & 0 & 726 & 0 \\ 0 & 0 & 1 & 0 & 0 & 2 & 0 & 0 & 0 & 286 \end{pmatrix}$$

4 Risultati sperimentali

4.1 Attacco non target

Abbiamo valutato l'**accuracy**¹ prima e dopo l'attacco su un batch di 100 immagini del test set attraverso la tecnica dell'attacco *non target* effettuato con tecnica *FGSM*.

Si ottiene che l'accuracy prima dell'attacco è pari a 91%, invece dopo l'attacco scende al 39%: da ciò si può dedurre che l'attacco funziona in maniera efficiente, con un success rate superiore al 50%.

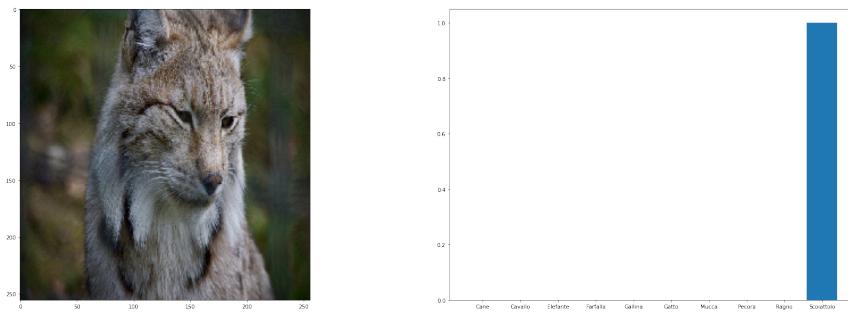


Figura 2: Esempio di attacco non target

4.2 Attacco non target iterativo

La versione iterativa dell'attacco si è rivelata ben più efficace in quanto, applicando perturbazioni con $\epsilon = 0.05$ e dunque alterando le immagini in maniera pressocchè impercettibile, l'accuracy della rete scende drasticamente. Per testare l'efficacia dell'attacco, abbiamo inizialmente testato l'accuracy su un batch di 32 immagini al variare del numero di iterazioni, con un valore di ϵ fissato e pari a 0.05. Come si nota dal grafico in figura 3, al crescere delle iterazioni il rumore dell'immagine aumenta e quindi è più facile che la rete vada in errore, facendo diminuire l'accuracy.

Abbiamo poi studiato l'accuracy con un numero di iterazioni fisso e pari a 10 al variare dell'ampiezza della perturbazione ϵ in un range compreso tra 0 e 0.30. Analizzando il grafico in figura 4, ci si rende subito conto dell'efficacia dell'attacco, per un rumore pari a $\epsilon = 0.06$ l'accuracy della rete passa dal 90% allo 0%.

¹L'accuracy è la percentuale media delle volte in cui la rete riesce a classificare in modo corretto un'immagine.

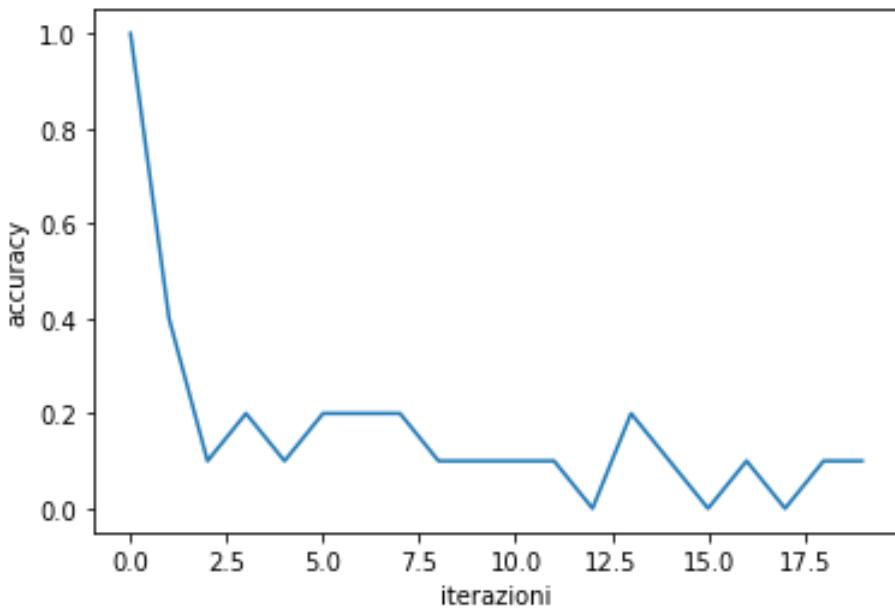


Figura 3: Accuracy al crescere delle iterazioni

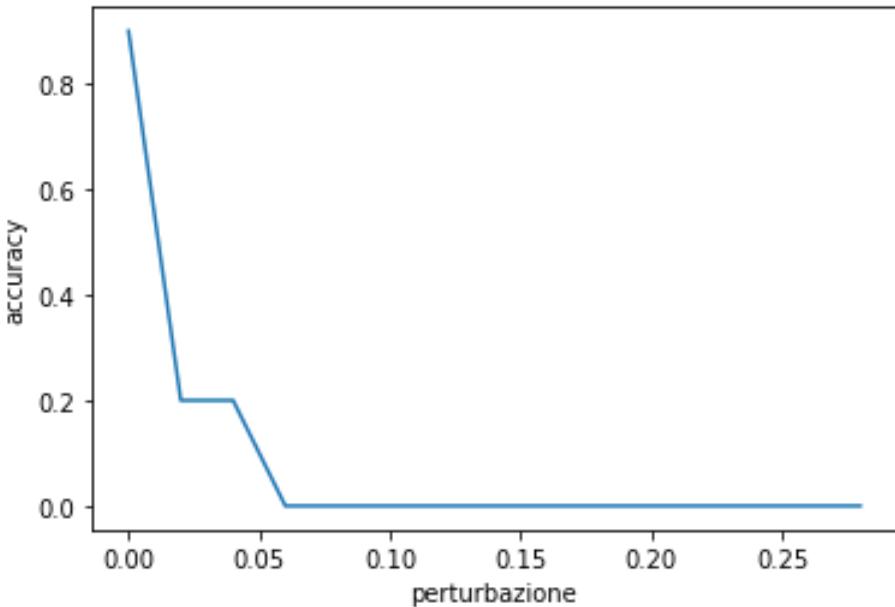


Figura 4: Accuracy al crescere della perturbazione

4.3 Attacco target iterativo

Abbiamo testato l'attacco *target* su cinque coppie di classi a scelta per verificarne l'efficacia. I risultati sono nel complesso positivi visto che la maggior

parte delle immagini viene confusa dalla rete neurale come appartenente ad un'altra classe. Per esempio, alla figura 7, una mucca, viene applicata una perturbazione impercettibile tale da essere confusa con un ragno, nonostante siano due categorie di animali totalmente diverse. Ad ogni immagine perturbata abbiamo allegato il relativo grafico a barre che mostra come la rete classifica quell'immagine.

Osservando i grafici, si può dire che l'attacco viene effettuato con successo 4 volte su 5: infatti, l'unico esperimento fallito è stato quello in figura 6, dove uno scoiattolo doveva essere confuso come elefante ma per via del poco rumore applicato all'immagine, la rete ha continuato a classificarlo come scoiattolo.

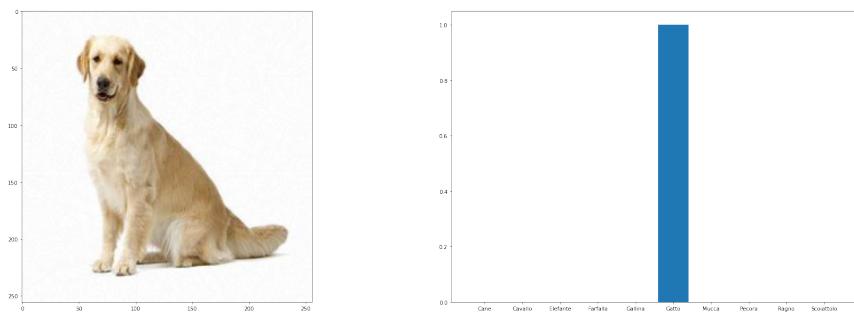


Figura 5: Esempio di attacco target, l'immagine originariamente classificata in maniera corretta classificata come appartenente alla classe scelta dall'attaccante, in questo caso un gatto

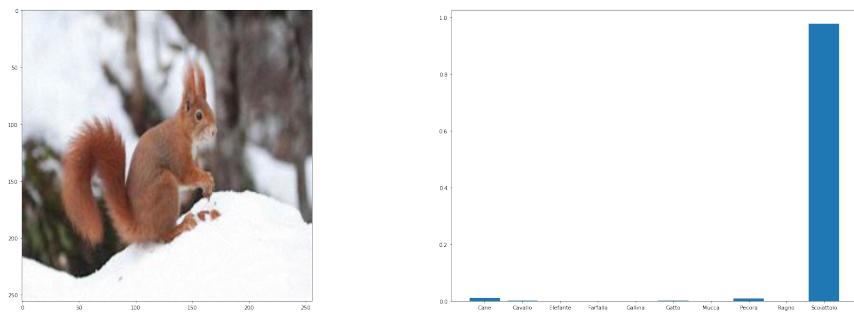


Figura 6: Esempio di attacco target fallito



Figura 7: Esempio di attacco target

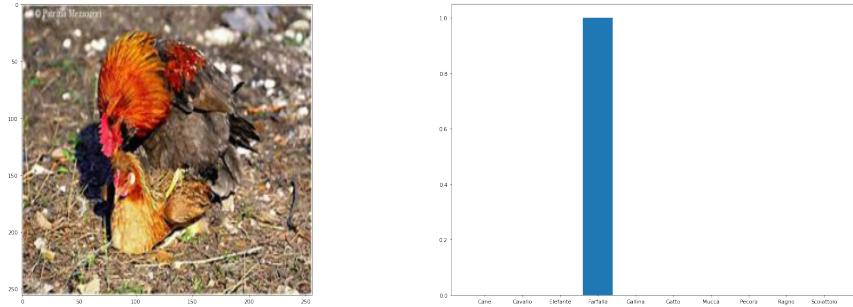


Figura 8: Esempio di attacco target

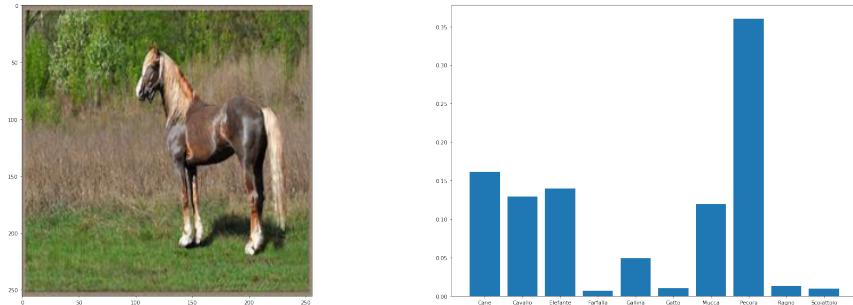


Figura 9: Esempio di attacco target

4.4 Attacco black box

Abbiamo valutato l'effetto dell'attacco *black box* con tecnica *FGSM non targeted iterativa* su un batch del dataset dedicato per il testing. Il numero di iterazioni dell'attacco e l'ampiezza della perturbazione sono stati scelti sperimentalmente al fine di ottenere le migliori prestazioni possibili. In questo esperimento, abbiamo misurato il **success rate**, ovvero la percentuale media di volte in cui la predizione effettuata dal modello black box sull'immagine originale è diversa da quella effettuata sull'immagine perturbata, consentendoci di capire quanto funziona l'attacco progettato. Su un batch di 10

immagini e ϵ uguale a 0.15, si riesce ad ottenere un success rate mediamente pari a 50%, un discreto risultato considerando che in questa tipologia di attacco non conosciamo la rete da attaccare. Come esempio campione, mostrato in figura 10, abbiamo analizzato l'immagine di un gatto, che, dopo essere stata perturbata, viene classificata dalla rete target come una pecora. Infine, è da notare la quantità di rumore applicata all'immagine confrontata con le immagini perturbate negli altri attacchi: da ciò si può evidenziare la **maggior complessità** degli attacchi black box, che non sono così affidabili come quelli white box a parità di rumore.

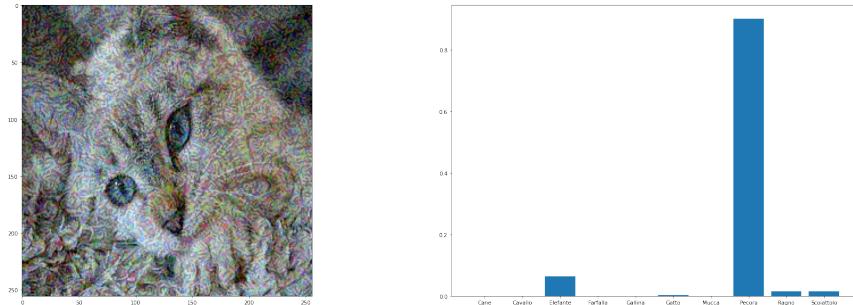


Figura 10: Immagine rumorosa di un gatto, classificato come pecora

References

- [1] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.
- [2] Zhangzhang Si and Song-Chun Zhu. Learning hybrid image templates (hit) by information projection. *IEEE Transactions on pattern analysis and machine intelligence*, 34(7):1354–1367, 2011.