



Adversarial perturbation detection

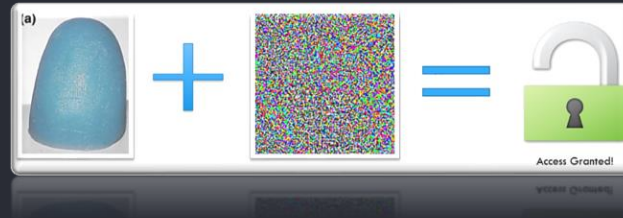
Corso di Data Mining

Studente: Di Chiara Guido
Matricola: M63000986

Formalizzazione del problema

I sistemi di **liveness detection** per impronte digitali consentono di distinguere se gli esemplari biometrici posti in ingresso ad essi sono autentici o rappresentano delle repliche artificiali.

Se il liveness detector si basa su una CNN, è possibile addestrare una GAN in grado di generare del rumore sulle acquisizioni delle impronte tale da eludere il sistema di controllo. Le modifiche apportate alle acquisizioni vengono dette **adversarial perturbation**.



L'obiettivo dello studio è quello di addestrare un classificatore in grado di discriminare le immagini di impronte digitali autentiche da quelle perturbate.

Soluzione proposta



Per risolvere il problema di classificazione binaria in questione, si è optato per una soluzione basata su CNN.

La scelta di tale soluzione si è basata sulla possibilità di mettere in pratica ed approfondire i concetti appresi nell'ambito degli approcci deep. Inoltre, i risultati derivati dall'utilizzo di questa tecnica hanno contribuito a confermare la validità della soluzione proposta.

Processo di risoluzione

1. **Analisi e preparazione dei dati**
Studio e partizionamento del dataset fornito per la generazione del modello
2. **Modellazione**
Selezione dell'architettura della rete e degli iperparametri da utilizzare
3. **Valutazione delle prestazioni**
Analisi della classificazione fornita dal modello sui dati di test
4. **Training del modello finale**
Addestramento del modello da utilizzare per predire le istanze non etichettate



1

Analisi e preparazione dei dati

Analisi dei dati forniti

Il dataset è composto dalle impronte rilevate da **190 soggetti**. Per ciascuno di essi si possiedono **10 acquisizioni** di impronte digitali genuine.

Per **182** di questi soggetti si possiedono altre 10 acquisizioni di impronte ognuno, le quali presentano adversarial perturbation.

Le acquisizioni sono all'incirca **bilanciate** tra le due categorie.

In totale si hanno dunque:

- 1900 acquisizioni «**Clean**»
- 1820 acquisizioni «**Adversarial**»

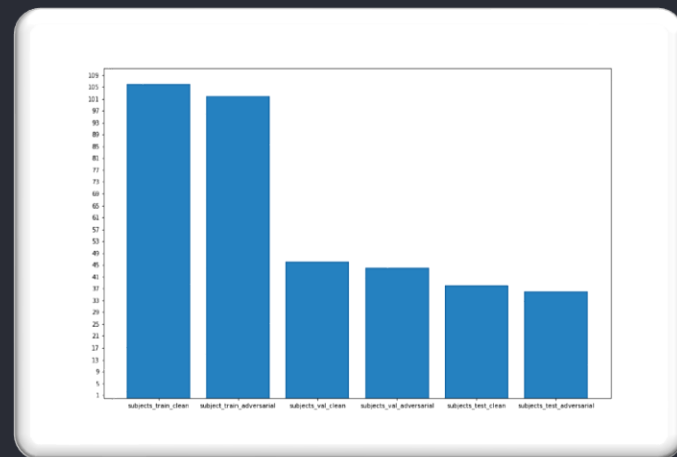


Partizionamento del dataset

Con una tecnica di **Holdout** sono stati creati tre sottoinsiemi, ciascuno dei quali costituito da tutte le acquisizioni relative agli stessi soggetti.

Le percentuali del dataset complessivo assegnate alle partizioni tenendo conto della **stratificazione** sono le seguenti:

- **Test set:** 20% delle istanze complessive
 - 38 soggetti Clean
 - 36 soggetti Adversarial
- **Training-Validation set:** 80% delle istanze complessive
 - **Training set:** 70% delle istanze
 - 106 soggetti Clean
 - 102 soggetti Adversarial
 - **Validation set:** 30% delle istanze
 - 46 soggetti Clean
 - 44 soggetti Adversarial





2

Modellazione

Strumenti utilizzati

La soluzione del problema è stata realizzata utilizzando i seguenti tool:



Linguaggio di
programmazione



Framework per
Deep learning



Ambiente di
esecuzione




Architettura della rete

Come architettura di rete è stato scelto il modello **AlexNet**. Tale iperparametro strutturale è stato fissato a valle di una fase di **model selection**, nella quale sono state testate le prestazioni di diverse architetture tra cui **VGG19** e **ResNet50**.

Le prestazioni offerte dalle reti sono state comparate dopo aver provato un set ridotto di configurazioni di iperparametri per un numero limitato di epoche.

L'architettura AlexNet ha ottenuto le migliori prestazioni in termini di **accuracy** sul validation in un minor numero di epoche rispetto alle alternative.

Inoltre, tale rete rappresenta il giusto compromesso tra capacità del modello e complessità del problema da risolvere. Infatti, gli altri modelli presentavano evidenti comportamenti di **overfitting** sui dati di training.



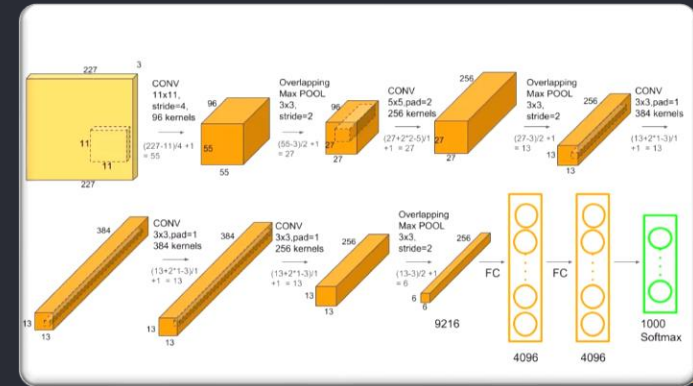
Adattamenti strutturali

L'architettura originale è stata adattata alle esigenze del problema in esame andando a sostituire l'**output layer** con un livello fully connected composto da soli due neuroni.

La funzione di attivazione dell'ultimo livello è stata inoltre ridefinita e si è scelto di utilizzare la versione logaritmica della **SoftMax**:

$$\text{LogSoftMax}(x_i) = \log \left(\frac{e^{x_i}}{\sum_j e^{x_j}} \right)$$

Tale versione perde l'interpretabilità del risultato in termini di probabilità ma guadagna stabilità numerica.



Strategie di aggiornamento dei pesi

Per quanto concerne la funzione di errore, si è scelto di utilizzare la **CrossEntropyLoss**:

$$CELoss(x, class) = - \frac{\log(e^{x[class]})}{\sum_j e^{x[j]}}$$

Come algoritmo di ottimizzazione si è preferito l'algoritmo adaptive moment estimation (**Adam**).

Infine, come tecnica di addestramento, si è scelto di optare per il **transfer learning** utilizzando la rete preaddestrata sul dataset **ImageNet**. L'algoritmo di backpropagation è stato esteso a tutti i livelli della rete.

Preprocessing e data augmentation

Prima di procedere con l'addestramento della rete, sono state applicate le seguenti trasformazioni alle immagini del **training set**:

- **Ridimensionamento:** necessario per adattare le dimensioni delle immagini a disposizione alla dimensione accettata dalla rete.
- **Flip orizzontale:** utilizzato come forma di **data augmentation**. Ogni immagine è stata ribaltata con una probabilità del 50%. Sperimentalmente si è osservato essere l'unica trasformazione necessaria per migliorare le prestazioni della rete.
- **Conversione in tensore:** necessario per poter utilizzare l'immagine come ingresso della rete.
- **Normalizzazione del tensore:** utilizzato per riscaldare a media nulla e varianza unitaria i valori dei tensori rappresentanti le immagini. I valori di riscaldamento (media e varianza) per ognuno dei tre canali sono relativi alle statistiche del dataset ImageNet su cui la rete è stata preaddestrata.

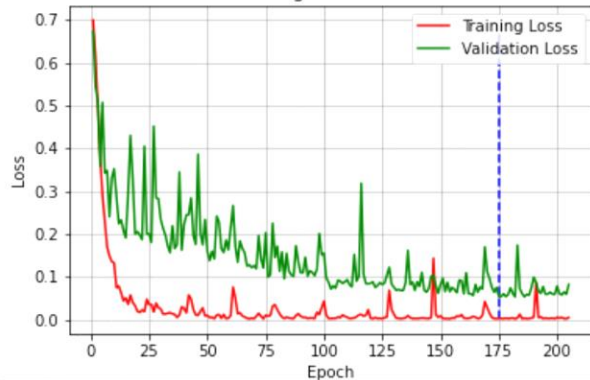
Tuning degli iperparametri

La fase di tuning degli iperparametri è stata condotta con un approccio **trial-and-error**, osservando il comportamento della rete sul **validation set**.

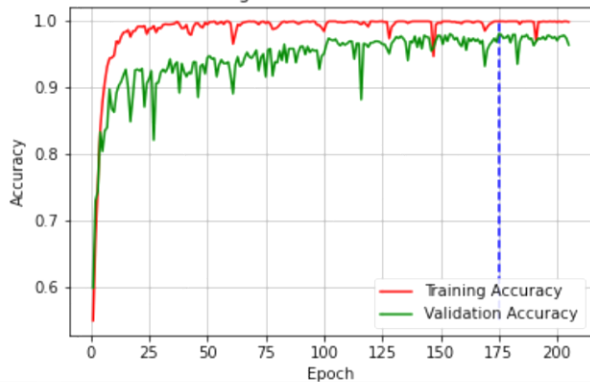
In particolare, oltre agli iperparametri strutturali discussi in precedenza e alle funzioni di attivazione e loss, sono stati fissati i seguenti valori:

- **Learning rate:** $9 \cdot 10^{-5}$
- **Weight decay:** 0.01
- **Batch size:** 32
 - Con shuffle del dataset ad ogni epoca

Model: Training Vs Validation Losses



Training Vs Validation Accuracies



Prestazioni sul validation set

Nei diagrammi a sinistra sono riportate le prestazioni ottenute per la migliore configurazione degli iperparametri.

Riaddestramento del modello

Osservando le prestazioni ottenute, si nota che dopo una fase di **apprendimento**, l'accuracy sul **validation** set si assesta. Questo è dovuto al fatto che avere un andamento circa costante della funzione di loss sul **training** set causa minime variazioni ai pesi del modello.

- Nel caso specifico, le prestazioni sul validation set sono estremamente elevate, pertanto non si sta manifestando overfitting.

Utilizzando la tecnica dell'**early stopping** Si è scelto di arrestare l'addestramento nel momento in cui l'accuracy sul training set non subisce miglioramenti per 30 epoche consecutive. L'epoca target evidenziata anche nei grafici è la 175

Stabilita l'epoca ideale, l'ultimo iperparametro tra quelli considerati viene fissato. Il modello è stato dunque riaddestrato sull'unione di training set e validation set per consentire la valutazione delle prestazioni sul test set indipendente.



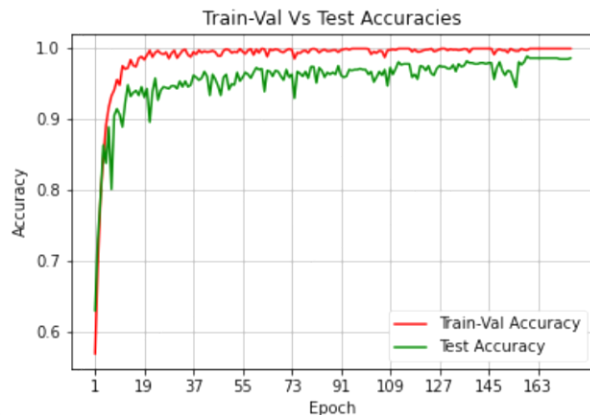
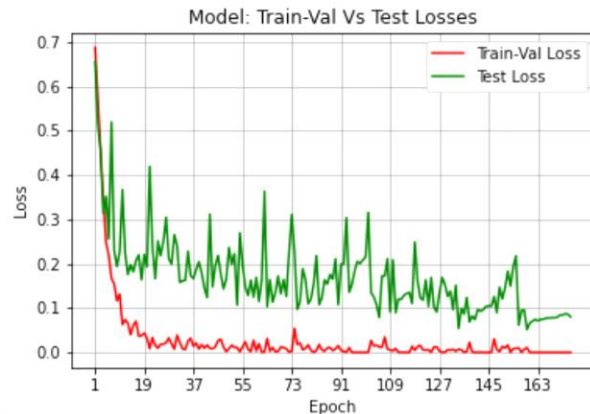
3

Valutazione delle prestazioni

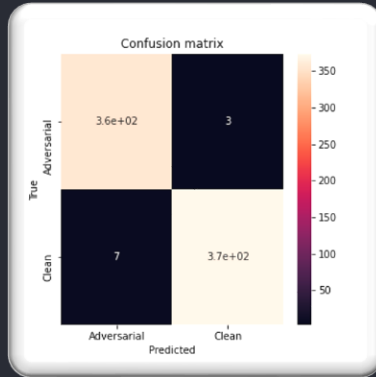
Prestazioni sul test set

Nei diagrammi a destra sono riportate le prestazioni ottenute dal modello sul test set.

Tali diagrammi sono stati ottenuti durante la fase di addestramento del modello.



Analisi dei risultati



Confusion matrix

I valori più significativi sono disposti lungo la diagonale principale, sinonimo di una corretta classificazione

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} = \frac{373 + 357}{373 + 357 + 7 + 3} = 0.987$$

$$FPR = \frac{FP}{TN + FP} = \frac{3}{357 + 3} = 0.008$$

Accuracy e False positive rate

L'accuratezza fornisce la capacità del modello di predire la classe delle istanze in ingresso.

L'FPR definisce il numero di previsioni «Clean» errate sul numero totale di «Adversarial»



4

Training del
modello finale

Modello definitivo

Valutate le prestazioni del modello sul test set, ed avendo accertato che il modello fornisca buone prestazioni, si procede con un passo di riaddestramento dello stesso.

Questa volta i dati di training saranno composti da tutti i dati etichettati a disposizione, lasciando invariato il numero di epoche di addestramento.

Al termine dell'addestramento, si è utilizzato il modello ottenuto per classificare i dati non etichettati e per creare la sottomissione per la piattaforma **Kaggle**.

	Fingerprint	Class
0	testFingerprint_1050.png	Clean
1	testFingerprint_152.png	Adversarial
2	testFingerprint_298.png	Clean
3	testFingerprint_320.png	Adversarial
4	testFingerprint_1163.png	Clean
...
1195	testFingerprint_138.png	Adversarial
1196	testFingerprint_1178.png	Adversarial
1197	testFingerprint_201.png	Adversarial
1198	testFingerprint_1067.png	Clean
1199	testFingerprint_315.png	Clean


1200 rows × 2 columns

1500 rows × 5 columns

1198 testFingerprint_315.png Clean

1198 testFingerprint_1067.png Clean



#	Team Name	Notebook	Team Members	Score ⓘ
1	Guido Di Chiara			1.00000

La classificazione prevede l'etichettatura di **1200 istanze**, per un 10% delle quali, viene mostrato nella **public leaderboard** lo score registrato dalla soluzione proposta.

Grazie per
l'attenzione!

