# M.I.A e I.I.A

I modelli di machine learning possono presentare dei rischi per la privacy nel caso in cui facciano trasparire delle informazioni sensibili sui dati di training attraverso il loro output. Si parla di Membership Inference Attack se è possibile con una certa accuratezza determinare se un certo dato fosse presente nel training set di un modello. Si parla invece di Identity Inference Attack se è possibile determinare se nel training dataset fosse presente qualche dato corrispondente ad una certa persona avendo a disposizione un altro dato della stessa persona (ad esempio a partire da una foto di un individuo determinare se nel training dataset fosse presente un'altra foto dello stesso individuo).

# Privacy nei modelli di machine learning

Dobbiamo definire cosa si intende per privacy nel machine learning: Una definizione possibile è quella che viene chiamata "differential privacy" ovvero si richiede che un attaccante non possa usare il modello per risalire a informazioni sul training dataset che non avrebbe potuto dedurre da un altro modello addestrato sulla stessa distribuzione del dataset. Questo quindi non significa che non si possa dedurre alcuna informazione sulla distribuzione dei dati di training. Infatti sarebbe una richiesta troppo forte in quanto un modello per essere utile deve generalizzare il più possibile le informazioni apprese dal training dataset a tutta la distribuzione che ha generato il dataset. Richiedere che dal modello non si possa dedurre alcuna informazione sulla distribuzione del dataset equivale a chiedere che il modello non funzioni. Differential privacy quindi è la richiesta che dal modello non si possa risalire a informazioni specifiche dei dati della distribuzione che erano presenti nel training dataset. Ad esempio un modello che identifica che una certa categoria di persone è predisposta ad una certa malattia non viola la differential privacy se:

1. È possibile costruire un altro modello tale che addestrandolo sulla stessa distribuzione troverà la stessa relazione tra categoria di persone e malattia.

2. Non è possibile ricavare informazioni sulle persone presenti nel training dataset, quindi il modello deve essere resistente a M.I.A. e I.I.A.

In questo esempio si nota che violare la differential privacy è particolarmente grave in quanto permette di accedere allo stato di salute di una specifica persona se presente nel dataset di addestramento.

[[Comprehensive_Privacy_Analysis_of_Deep_Learning_Passive_and_Active_White-box_Inference_Attacks_against_Centralized_and_Federated_Learning.pdf]] (qua evidenziato in rosa e anche nell'altro paper di Reza) TODO da mettere a posto

# Black Box e White Box

Per gli inference attacks possiamo usare due paradigmi diversi.

- White box (scatola bianca) ovvero l'attaccante conosce informazioni sul modello da attaccare come: tipo di modello utilizzato, numero di layer, parametri.

(il paper di Zhang sfrutta federated learning per ottenere dei parametri e' quasi white box TODO)

- Black box (scatola nera) ovvero l'attaccante non ha alcuna informazione sul funzionamento interno del modello e può solo utilizzarlo osservando gli output forniti.

Il modello black box può essere visto come una API a cui è possibile fare delle richieste con degli input scelti dall'attaccante, se richiesti dal modello target, e ottenere le risposte da utilizzare per l'attacco. Ci concentreremo principalmente su questo secondo modello in quanto è più simile ad un caso reale in cui un attaccante effettua un M.I.A. su un modello target a cui non ha accesso direttamente.

# M.I.A. su classificatori

Chiamiamo $\mathcal{T}$ il modello target su cui vogliamo svolgere l'attacco.
Chiamiamo $\mathcal{D}_{train,\mathcal{T}}$ il dataset di training di $\mathcal{T}$.

Nei modelli di machine learning di classificazione, il modello determina a quale tra $k$ classi è più probabile appartenga l'input. Il classificatore dà in output un vettore lungo $k$ dove ogni componente rappresenta la probabilità che l'input appartenga alla corrispondente classe. Ad esempio $\begin{bmatrix} cane \\ gatto \\ orso \\ volpe \end{bmatrix} = \begin{bmatrix} 0.6 \\ 0.1 \\ 0.1 \\ 0.2 \end{bmatrix}$ L'intuizione su cui ci basiamo è che il modello classificherà in maniera diversa input che erano già presenti nel training set ($\mathcal{D}_{train,\mathcal{T}}$) con una confidenza maggiore nel vettore di predizione. Ad esempio un cane presente nel training set viene classificato con alta confidenza: $\begin{bmatrix} cane \\ gatto \\ orso \\ volpe \end{bmatrix} = \begin{bmatrix} 0.9 \\ 0.025 \\ 0.025 \\ 0.05 \end{bmatrix}$ Come analizzato dal Paper di REZA(aggiungi link TODO) l'overfitting del modello target $\mathcal{T}$ rende maggiori le differenze nella confidenza della classificazione tra dati nuovi e dati già visti dal modello durante il training. Quindi l'overfitting di $\mathcal{T}$ facilita attacchi di inferenza.

Possiamo quindi addestrare un nuovo modello di machine learning $M_{inference}$, un classificatore binario che a partire da queste differenze nell'output tra i dati in $\mathcal{D}_{train,\mathcal{T}}$ e quelli in $\overline{\mathcal{D}_{train,\mathcal{T}}}$ (il complemento) determini se l'input $\in \mathcal{D}_{train,\mathcal{T}}$ o no.

Per poter addestrare $\mathcal{M}_{inference}$ avremmo bisogno dei vettori di predizione (e.g. $\begin{bmatrix} cane \\ gatto \\ orso \\ volpe \end{bmatrix}$) con la corrispondente label `in` o `out` in base all'appartenenza a $\mathcal{D}_{train,\mathcal{T}}$.

Non abbiamo a disposizione questi dati per il modello $\mathcal{T}$, quindi creiamo una serie di "shadow models" $\mathcal{S}_i$ il cui scopo è imitare $\mathcal{T}$. Siccome questi $\mathcal{S}_i$ sono creati dall'attaccante può controllarne il training set $\mathcal{D}_{train,\mathcal{S}_i}$ e quindi ha a disposizione dei vettori di predizione con la corrispondente label `in` e `out`. Possiamo quindi addestrare il classificatore binario $\mathcal{M}_{inference}$ in modo che determini se un certo input appartenga a $\mathcal{D}_{train,\mathcal{S}_i}$ oppure no in base al vettore di predizione corrispondente. L'idea è che se gli shadow models $\mathcal{S}_i$ si comportano in maniera abbastanza simile a $\mathcal{T}$ la capacità di $\mathcal{M}_{inference}$ di discriminare `in` e `out` su dati in $\mathcal{D}_{train,\mathcal{S}_i}$ si tradurrà nella capacità di

discriminare `in` e `out` su dati in $\mathcal{D}_{train,\mathcal{T}}$. In questo caso avere informazioni aggiuntive su $\mathcal{T}$ (ad esempio il tipo del modello) permetterebbe di creare dei $\mathcal{S}_i$ più simili aumentando l'accuratezza del nostro attacco. Per l'addestramento dei $\mathcal{S}_i$ avremo che $\mathcal{D}_{train,\mathcal{S}_i}$ contiene tutti dati con una determinata classe $i$ (ad esempio $\mathcal{D}_{train,\mathcal{S}_i}$ ha tutti cani,$\mathcal{D}_{train,\mathcal{S}_2}$ ha tutti gatti ecc...), questo perché la distribuzione del vettore di predizione può dipendere dalla classe dell'input (ad esempio se è più facile per un modello classificare certi animali rispetto ad altri). Quindi si addestrano $k$ shadow models uno per ogni classe in modo da catturare la distribuzione dei vettori di predizione condizionata dalla appartenenza ad una classe. Per il modello $\mathcal{M}_{inference}$ si può utilizzare qualsiasi modello che permetta la classificazione binaria.

---

## creazione dataset shadow

come creiamo i dataset shadow TODO se serve Guardare da tutti i paper per le varie spiegazioni specialmente Reza e GANMIA.

---

# GAN

Le generative neural network sono una classe di modelli di deep learning utilizzati per generare dei dati dalla stessa distribuzione dei dati di training. A differenza di altri modelli generativi le GAN non approssimano esplicitamente la densità di distribuzione dei dati di training ma trasformano del rumore in input in dei dati con una alta likelihood di appartenere alla distribuzione dei dati di training. Una GAN è composta da due reti neurali: una rete generatrice che prende in input rumore e genera dei dati e una rete discriminatrice che prende in input i dati provenienti dalla distribuzione originale oppure quelli generati dalla rete generatrice e cerca di determinare se siano dati originali o generati. La rete discriminatrice viene addestrata in modo da raggiungere la massima probabilità che riesca a distinguere dati originali da quelli generati. La rete generatrice viene addestrata in modo da massimizzare la probabilità che la rete discriminatrice classifichi i dati generati come dati originali. Le due

reti vengono addestrate alternandosi cercando di tenere la rete discriminatrice vicino all'ottimalità in modo da forzare la rete generatrice a generare dati più simili a quelli originali.

## Value function della GAN

Dati:

- $D(x)$ è la probabilità che il discriminatore classifichi correttamente un dato proveniente dalla distribuzione originale.
- $D(G(z))$ è la probabilità che il discriminatore classifichi erroneamente un dato generato come provieniente dalla distribuzione originale.

Una GAN può essere modellata come un gioco minimax dove il generatore $G$ e il discriminatore $D$ competono con la seguente value function: $\min_G \max_D \{\underbrace{\mathbb{E}_x[\log D(x)]}_{\bigstar} + \underbrace{\mathbb{E}_z[1 - \log D(G(z))]}_{\blacklozenge}\}$

Il generatore può intervenire solo su $\blacklozenge$. Minimizzando $\blacklozenge$ il generatore sta diminuendo la probabilità(TODO forse sarebbe corretto scrivere likelihood???) che $D$ riesca a distinguere gli output di $G$ da dati reali. Quindi $G$ cerca di aumentare i falsi positivi di $D$.

Il discriminatore cerca di massimizzare $\blacklozenge$ ovvero la probabilità (TODO forse va likelihood) di identificare come falsi i dati generati da $G$. Quindi $D$ cerca di diminuire i falsi positivi. Inoltre $D$ massimizza $\bigstar$ ovvero la probabilità (TODO forse likelihood) che classifichi come veri i dati provenienti dalla distribuzione originale. Quindi $D$ cerca di diminuire i falsi negativi.

Il termine $\bigstar$ nella value function è necessario per evitare che $D$ classifichi come falso ogni input, ottenendo precisione del 100% ma recall del 0%.

È stato dimostrato nel paper di Ian Goodfellow (TODO link) che alternando il gradient descent di $D$ e $G$ la GAN converge a $D(x) = \frac{1}{2}$ ovvero il generatore crea dei dati che $D$ non riesce a distinguere da quelli originali.

# M.I.A su GAN

(TODO mettere link al paper di Hayes) Un M.I.A. su una GAN presenta delle difficoltà ulteriori rispetto a quello su un modello classificatore. Infatti per l'attacco su un classificatore si hanno a disposizione i vettori di predizione. L'attaccante può sfruttare i diversi livelli di confidenza nei vettori di predizione su input appartenenti ai dati di training rispetto a dati mai visti dal classificatore per determinare se un certo dato appartenente al training set.

## paradigma white box

Nel paradigma white box contro una GAN si ha disposizione la rete neurale discriminatrice $D$ utilizzata durante il training. Per l'attacco è sufficiente utilizzare la rete discriminatrice e, nel caso di overfitting, gli input appartenenti al training dataset avranno una confidenza più alta quando vengono classificati. Questo procedimento è simile al M.I.A. su classificatori che utilizza la confidenza del vettore di predizione.

## paradigma black box senza alcuna informazione addizionale

Nel paradigma di attacco black box senza informazioni addizionali su una GAN non abbiamo a disposizione gli output delle rete neurale discriminatrice, abbiamo solo a disposizione i dati generati dalla GAN.

L'idea per poter effettuare un attacco è di addestrare localmente una GAN. Si può sfruttare l'attacco white box sulla GAN locale di cui abbiamo a disposizione la rete discriminatrice. Se la GAN locale è abbastanza simile a quella da attaccare e quest'ultima ha overfitting possiamo con successo determinare se un dato era presente nel training set. Questa idea è simile a quella degli shadow models utilizzati per il M.I.A. nei classificatori: creare un modello locale di cui l'attaccante ha il controllo per simulare il modello target che vuole attaccare. Per addestrare la GAN locale non si hanno a disposizione membri del traning dataset della GAN attaccata quindi si usano i dati generati da quest'ultima.

## black box con informazioni addizionali

Il paradigma precedente dove l'attaccante non ha nessuna informazione sul modello da attaccare è molto restrittiva, a volte l'attaccante ha a disposizione una parte del dataset originale (una parte del dataset è pubblica, data breach, foto o testi presi da internet ecc.). Ad esempio per una GAN che genera testi l'attaccante può sapere che nel training dataset fosse presente un certo romanzo e voler inferire altri testi usati nell'addestramento. L'attaccante può sfruttare queste informazioni aggiuntive sul training dataset per migliorare le prestazioni dell'attacco in due modi: attacco discriminativo e attacco generativo.

Nell'attacco discriminativo è richiesto che l'attaccante abbia a disposizione: Alcuni dati che non siano stati utilizzati nel training della GAN, che chiamiamo $\mathcal{A}_{\text{not train}}$, ed eventualmente alcuni dati appartenenti al training set, chiamati $\mathcal{A}_{\text{train}}$. Si addestra una rete discriminatrice locale dove come input falsi vengono dati $\mathcal{A}_{\text{not train}}$ e come input veri dei dati generati dalla GAN target più, se sono a disposizione dell'attaccante, anche i dati di $\mathcal{A}_{\text{train}}$. In questo modo il discriminatore locale impara a differenziare dati non presenti nel training set da quelli presenti nel training set o generati dalla GAN target. Se la GAN target presenta overfitting danno informazioni sul training set. Una volta addestrato il discriminatore locale si può procedere con un attacco uguale a quello white box.

Nell'attacco generativo è richiesto che l'attaccante abbia a disposizione: Alcuni dati che siano stati utilizzati nel training della GAN, che chiamiamo $\mathcal{A}_{\text{train}}$, ed eventualmente alcuni dati appartenenti al test set, chiamati $\mathcal{A}_{\text{test}}$. Si addestra una GAN locale dove come input veri vengono usati $\mathcal{A}_{\text{train}}$ e dati generati dalla GAN target mentre come input falsi vengono usati dati generati dalla GAN target e, se sono a disposizione, anche i dati di $\mathcal{A}_{\text{train}}$. Di nuovo si può usare il discriminatore della GAN locale per procedere con un attacco white box.

È da notare come nell'approccio discriminativo sia necessario avere $\mathcal{A}_{\text{not train}}$ ma $\mathcal{A}_{\text{train}}$ potrebbe essere vuoto, in questo caso si possono anche usare solo i dati generati dalla GAN target. Nell'approccio generativo invece è $\mathcal{A}_{\text{test}}$ a non essere necessario. In ogni caso avere più informazioni possibili sia sul training che sul test dataset originale migliorerà le prestazioni dell'attacco.

(TODO nelle altre versioni tipo GANMIA i requisiti sono diversi).

---

## Precisione VS Recall nei Membership inference attacks

TODO da mettere se ha senso