Università degli Studi di Bari “Aldo Moro”

Dipartimento di Informatica

Corso di laurea in Informatica e Tecnologie per la Produzione del Software

Esame di

**Modelli e Metodi per la Sicurezza delle Applicazioni**

Sperimentazione di differenti Attacchi a Reti Neurali

Professore:

**Dott. Donato** **Impedovo**

Dottorando:  
**Dott. Francesco** **Castro**

Realizzato da:   
**Nicola** **Balzano**

Anno Accademico 2023/2024

Sommario

[1 Introduzione 6](#_Toc168386518)

[2 Stato dell’Arte 7](#_Toc168386519)

[2.1 Tassonomia 7](#_Toc168386522)

[2.2 Metodologia di attacco 9](#_Toc168386523)

[2.2.1 Poisoning 9](#_Toc168386524)

[2.2.2 Backdoor Poisoning 10](#_Toc168386525)

[2.3 Fast Gradient Sign Method 11](#_Toc168386526)

[2.4 BadNets 13](#_Toc168386527)

[2.5 Clean Label 14](#_Toc168386528)

[2.6 WaNet 16](#_Toc168386529)

[2.7 Hidden Trigger Backdoor 18](#_Toc168386530)

[3 Dataset 21](#_Toc168386531)

[4 Metodi 22](#_Toc168386533)

[4.1 Rete neurale utilizzata 22](#_Toc168386535)

[4.1.1 Spiegazione della rete neurale 22](#_Toc168386536)

[4.1.2 Parchè lavorare sulla scala dei grigi? 24](#_Toc168386537)

[4.2 Utilizzo Di FGSM 25](#_Toc168386538)

[4.3 Utilizzo di BadNets 25](#_Toc168386539)

[4.4 Utilizzo di Clean Label 26](#_Toc168386540)

[4.5 Utilizzo di WaNet 26](#_Toc168386541)

[4.6 Utilizzo di Hidden Trigger Backdoor 26](#_Toc168386542)

[5 Sperimentazione 27](#_Toc168386543)

[5.1 Protocollo seguito 27](#_Toc168386545)

[6 Risultati 30](#_Toc168386546)

[6.1 Modello originale 30](#_Toc168386548)

[6.2 FGSM 33](#_Toc168386549)

[6.2.1 Test con = 0.01 33](#_Toc168386550)

[6.2.2 Test con = 0.05 33](#_Toc168386551)

[6.2.3 Test con = 0.1 34](#_Toc168386552)

[6.2.4 Test senza label target 34](#_Toc168386553)

[6.2.5 Conclusione 34](#_Toc168386554)

[6.3 BadNets 35](#_Toc168386555)

[6.3.1 Trigger 5x5 36](#_Toc168386556)

[6.3.2 Trigger 2x2 37](#_Toc168386557)

[6.3.3 Trigger 7x7 37](#_Toc168386558)

[6.3.4 Conclusioni 38](#_Toc168386559)

[6.4 Clean Label 39](#_Toc168386560)

[6.4.1 Trigger 2x2 39](#_Toc168386561)

[6.4.2 Trigger 5x5 40](#_Toc168386562)

[6.4.3 Trigger 7x7 41](#_Toc168386563)

[6.4.4 Conclusioni 41](#_Toc168386564)

[6.5 WaNet 42](#_Toc168386565)

[6.5.1 s = 2 43](#_Toc168386566)

[6.5.2 s = 5 44](#_Toc168386567)

[6.5.3 s = 7 44](#_Toc168386568)

[6.5.4 s = 9 45](#_Toc168386569)

[6.5.5 Conclusioni 45](#_Toc168386570)

[6.6 Hidden Trigger Backdoor 47](#_Toc168386571)

[6.6.1 Trigger 2x2 47](#_Toc168386572)

[6.6.2 Trigger 5x5 48](#_Toc168386573)

[6.6.3 Trigger 7x7 48](#_Toc168386574)

[6.6.4 Cocnlusione 49](#_Toc168386575)

[6.7 Considerazioni finali 51](#_Toc168386576)

[7 Bibliografia 53](#_Toc168386577)

1. Introduzione

Il crescente utilizzo dei modelli di machine e deep learning in applicazioni critiche ha sollevato preoccupazioni riguardo alla loro sicurezza e robustezza. Mentre questi modelli sono in grado di raggiungere prestazioni straordinarie in una vasta gamma di compiti, dalla visione artificiale alla comprensione del linguaggio naturale, essi sono anche vulnerabili a vari tipi di attacchi. Gli attacchi possono compromettere la correttezza e l'affidabilità dei modelli, con potenziali conseguenze gravi.

Il seguente studio tratta della sperimentazione di vari attacchi ad un modello di machine learning che effettua la classificazione di CT scan COVID e non-COVID, al fine di comprendere le reali conseguenze e vulnerabilità di questi modelli quando non sono dotati di alcun meccanismo di protezione. In particolare, lo studio si sofferma su due categorie di attacchi: attacchi di avvelenamento (**poisoning**) e attacchi di avvelenamento con backdoor (**backdoor** **poisoning**). Questi attacchi sono stati selezionati per la loro capacità di compromettere l'integrità dei modelli di machine learning in modi subdoli e difficili da rilevare.

1. Stato dell’Arte

# Tassonomia

È possibile differenziare gli attacchi in ambito di **Adversarial Machine Learning** (**AML**) secondo **3 macro categorie** [1]:

* **Black-box**:Gli attacchi di questa categoria presuppongono che il threat agent/group[[1]](#footnote-1) non abbiano alcuna conoscenza interna del modello di machine learning, né dei suoi parametri o della sua architettura. Gli attaccanti possono solo osservare le risposte del modello agli input forniti.
* **White-box**:gli attacchi di questa categoria presuppongono che il threat agent/group siano in possesso di una piena conoscenza del sistema di ML.
* **Grey-box**: Gli attacchi di questa categoria si collocano tra i black-box e i white-box, dove il threat agent/group hanno una conoscenza parziale del modello di machine learning. Ad esempio, l'attaccante potrebbe conoscere l'architettura del modello ma non i valori esatti dei suoi parametri, i suoi pesi, o avere accesso a un sottoinsieme dei dati di training.

Nella tassonomia degli attacchi ai modelli di machine learning, oltre alla differenza tra white-box, black-box e grey-box e ai vari tipi di attacco, si possono anche classificare in base al target a cui mirano:

* + **Disponibilità**: Rendere il modello inutilizzabile o degradare le sue prestazioni.
  + **Integrità**: Modificare l'output del modello in modo malevolo, come nei backdoor attacks.
  + **Riservatezza**: Estrarre informazioni sensibili dal modello, come negli inference attacks.

Le principali tipologie di attacchi presenti a Stato dell’Arte [1] su reti non generative sono:

* **Evasion attacks**: Questi attacchi mirano a indurre il modello a fare previsioni errate, modificando leggermente i dati di input. Ad esempio, piccole perturbazioni in un'immagine possono portare un modello di riconoscimento a classificare erroneamente l'oggetto, manipolando quindi l**’integrità** del modello.
* **Backdoor Poisoning attacks:** Coinvolgono l'inserimento di dati malevoli nel processo di training del modello, compromettendo così la **qualità** e **integrità** del modello stesso.
* **Side-Channel Attacks**: Questi attacchi sfruttano informazioni indirette, come il consumo energetico, il tempo di esecuzione o le emissioni elettromagnetiche del sistema, per inferire dettagli sui modelli e sui dati con cui operano. Anche se non interferiscono direttamente con i dati, possono rivelare informazioni sensibili che possono essere utilizzate per ulteriori attacchi e possono vincolare la **disponibilità** del modello.
* **Model Inversion Attacks**: Questo tipo di attacco mira a ricostruire dati di input originali utilizzando l'accesso al modello. Gli attaccanti cercano di estrarre informazioni sensibili dai modelli, ricostruendo, ad esempio, immagini di volti o altri dati personali a partire dalle risposte del modello, violandola **privacy** e vincoli di **copyright**.
* **Model Extraction Attacks**: Gli attaccanti cercano di copiare o "rubare" un modello di machine learning interrogandolo ripetutamente con input accuratamente selezionati per ricostruire il modello stesso. Questo permette di ottenere un modello che funziona in modo simile a quello originale, compromettendo la **proprietà** **intellettuale**, la **sicurezza** **dei** **dati** e permette di definire un modello di base da cui partire per effettuare le altre tipologie precedentemente descritte.

# Metodologia di attacco

In questo studio, le metodologie di attacco scelte per testate la rete neurale sono il poisoning [1] e la sua versione con l’utilizzo di un backdoor “trigger” [1].

## Poisoning

Il **poisoning** [1] è un tipo di attacco in cui l'attaccante altera il dataset di addestramento inserendo esempi manipolati, senza l'uso di un trigger specifico. L'obiettivo è degradare la qualità del modello compromettendo la sua capacità di generalizzare correttamente su nuovi dati. In questo scenario, il modello viene addestrato su dati contaminati che rappresentano distribuzioni non corrette, portandolo a imparare associazioni errate e a fare previsioni sbagliate su dati puliti. Questi attacchi sono difficili da rilevare poiché i dati contaminati possono sembrare legittimi e coerenti con il resto del dataset.

## Backdoor Poisoning

A differenza di ciò il **backdoor** **poisoning** [1] comporta l'inserimento intenzionale di una percentuale di esempi manipolati all’interno del dataset di addestramento. Questi esempi sono progettati per far sì che il modello apprenda un comportamento specifico, che viene attivato da un "trigger" quando il modello è in uso, come una backdoor (“porta aperta”) all’interno del modello. Perciò quando il modello verrà testato con i dati che non presentano il trigger, funzionerà correttamente, mentre quando effettuerà delle predizioni su dati con il trigger verrà “attivata” la backdoor e farà una erronea predizione, in base alla target label definita dall’attaccante.

**Vantaggi degli attacchi descritti**

* **Difficile da Rilevare**: Poiché il trigger può essere molto piccolo o inserito in modo subdolo, il backdoor poisoning è spesso difficile da rilevare attraverso l'analisi tradizionale dei dati di addestramento.
* **Efficienza**: È necessario modificare solo una piccola percentuale del dataset per avere un impatto significativo sul comportamento del modello, mantenendo bassa la probabilità di rilevamento.

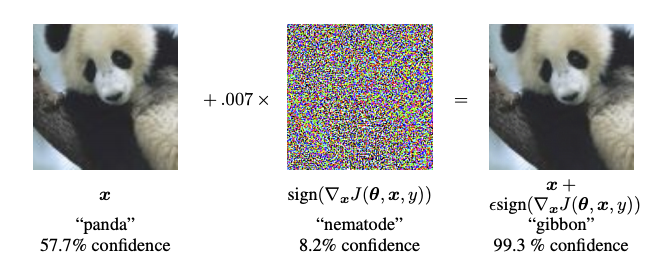
**Svantaggi degli attacchi descritti**

1. **Necessità di Accesso al Dataset di Addestramento**:
   * **Difficoltà di Accesso**: Per eseguire un attacco di backdoor poisoning, l'attaccante deve avere accesso al dataset di addestramento o al processo di addestramento del modello, il che può essere difficile in ambienti controllati o sicuri.
   * **Monitoraggio e Controlli**: I dataset di addestramento spesso sono monitorati e protetti, riducendo le possibilità che un attaccante possa inserirvi dati malevoli senza essere scoperto.
2. **Requisiti di Conoscenza del Modello**:
   * **Conoscenza white-box**: L'attaccante necessita di una buona comprensione del modello, del suo processo di addestramento e delle sue caratteristiche per creare un trigger efficace, il che richiede competenze tecniche avanzate.
3. **Possibili Difese Avanzate**:
   * **Adversarial Training**: Utilizzo di tecniche come l’adversarial training[[2]](#footnote-2) possono migliorare la robustezza del modello contro attacchi, rendendo più difficile l'efficacia del backdoor poisoning.

Dopo aver delineato la tassonomia degli attacchi di avvelenamento, è opportuno esaminare più da vicino alcune delle tecniche specifiche che saranno oggetto della nostra sperimentazione. Questi attacchi rappresentano alcune delle strategie più avanzate e insidiose utilizzate per compromettere i modelli in fase di addestramento.

# Fast Gradient Sign Method

Il **Fast Gradient Sign Method** [3] (FGSM) è una tecnica di attacco di evasion black box (figura 1), utilizzata per generare perturbazioni minime ma efficaci su un input al fine di ingannare un modello.

  
Figura 1: Esempio di applicazione del metodo FGSM

La prima fase dell'FGSM prevede il calcolo del gradiente della funzione di perdita del modello rispetto all'input originale. La funzione di perdita misura quanto la previsione del modello differisce dall'etichetta corretta .

Il **gradiente** viene rappresentato come:

dove sono i parametri del modello e è l’input originale.

Dopo aver calcolato il gradiente, si prende il segno di ciascun componente del gradiente, generando una direzione di perturbazione per ogni pixel dell'immagine di input. Il **segno** **del** **gradiente** indica la direzione in cui bisogna modificare ciascun pixel per aumentare la funzione di perdita. In altre parole, si decide semplicemente se aumentare o diminuire il valore di ogni pixel per aumentare la perdita.

* Se il gradiente di un pixel è **positivo**, il segno sarà +1, suggerendo di aumentare il valore di quel pixel.
* Se il gradiente di un pixel è **negativo**, il segno sarà -1, suggerendo di diminuire il valore di quel pixel.

La **perturbazione** viene poi applicata all'input originale aggiungendo una piccola quantità nella direzione del segno del gradiente. La formula per calcolare l'input perturbato è:

dove è un parametro di controllo che determina l'intensità della perturbazione. Un valore piccolo di assicura che la perturbazione sia impercettibile a occhio nudo.

A stato dell’arte FSGM viene utilizzata come tecnica di evasion per la sua semplicità di utilizzo e funzionamento, ma può comunque essere utilizzata come metodo di poisonig al fine di andare a perturbare i dati di training del modello avversario, non rendendo molto visibile l’attacco.

# BadNets

**BadNets** [4], il primo attacco di tipo backdoor poisoning black box, viene attuato durante l'addestramento del modello, dove in queta fase vengono utilizzati sia dati puliti che dati contenenti i trigger. Quando un'immagine con trigger[[3]](#footnote-3) viene presentata al modello durante l'addestramento, l'errore di classificazione risultante viene propagato all'indietro attraverso il modello utilizzando l'algoritmo di **retropropagazione**. Questo processo aggiorna i pesi del modello in modo che il modello possa fare previsioni più accurate anche su dati contenenti il trigger.

La presenza dei trigger nelle immagini di addestramento e le relative etichette errate causano una modifica nei pesi del modello che rende il modello sensibile ai trigger (figura 2).

A stato dell’arte questo è il primo attacco di backdoor poisoning documentato, risalente al 2017.

  
Figura 2: Esempio di utilizzo di BadNets in Traffic Sign Recognition, il cartello stradale con il post-it è stato associato alla label speedlimit

Lo **svantaggio** principale di questo attacco è che va ad ampliare il dataset del modello, tecnica conosciuta con il nome di **augmentation**, inserendo delle immagini di training con etichette erronee, il che è visibile all’occhio umano, anche se viene avvelenata solo una percentuale del dataset, il che diminuisce la probabilità di rilevamento.

# Clean Label

**Clean label** [5]è una tecnica di attacco backdoor poisoning white box sviluppata sulla base di BadNets [4], dove gli attaccanti manipolano i dati di addestramento in modo tale che le etichette associate alle immagini rimangano corrette (“pulite”), ma l'immagine stessa viene alterata in modo sottile. L'obiettivo è far sì che il modello impari una backdoor o un comportamento indesiderato senza che le etichette sembrino errate o sospette.

**Prima di applicare il trigger** alle immagini, vengono manipolate le immagini originali di training al fine di rendere quest’ultime più difficili da classificare così da incoraggiare il modello a memorizzare il pattern come una feature.

Come descritto in questa tecnica a Stato dell’Arte, verrà usato un Adversarial Trainer che utilizza il **Projected Gradient Descent** (**PGD**).

Cioè, per un classificatore , con funzione di Loss[[4]](#footnote-4) e input , costruiamo le immagini perturbate come:

per una certa norma e un limite .

Questa espressione descrive che la perturbazione aggiunta all’immagine originale è tale che l’immagine risultante sia visivamente indistinguibile dall’originale, cioè la norma della differenza delle due rispetta un certo limite. Viene quindi cercata tramite la funzione l’immagine perturbata che massimizza la funzione tra gli input che rispettano il vincolo sulla norma , generando così in modo iterativo il vettore delle immagini perturbate .

Come norma p si possono utilizzare varie norme, quelle testate a Stato dell’Arte sono la norma 2 e la norma infinito (inf). La **differenza** principale nell’utilizzo della **norma 2 o inf** (figura 3) è che, la norma 2 tende a distribuire uniformemente la perturbazione su tutti i pixel dell’immagine e possono quindi essere meno rilevabili rispetto a quelli generati con la norma inf, la quale enfatizza la perturbazione su pochi pixel.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  | Original img |
|  |  |  |  |

Figura 3: Esempio di applicazione delle norme al variare di

# WaNet

**WaNet** [6] è un attacco di tipo backdoor poisoning black box che introduce un trigger impercettibile nelle immagini di training per indurre un comportamento malevolo nel modello addestrato. A differenza degli attacchi di tipo "visible trigger" come BadNets [5], WaNet utilizza una mappa di deformazione (**warping** **grid**) per applicare una trasformazione geometrica alle immagini, rendendo il trigger più difficile da rilevare.

WaNet si basa sull'idea di creare una mappa di deformazione che, quando applicata alle immagini di input, provoca una leggera distorsione. Questa distorsione funge da trigger backdoor. Il modello addestrato con queste immagini avvelenate apprende a riconoscere queste distorsioni come una caratteristica del target label specificato dall'attaccante.

La mappa di deformazione è creata utilizzando una griglia che definisce come i pixel delle immagini devono essere spostati. Questo spostamento è definito in termini di una funzione continua e liscia che applica una lieve distorsione geometrica.

Matematicamente, sia 𝐼 l'immagine originale e 𝑊 la mappa di deformazione, allora l'immagine avvelenata 𝐼′ è ottenuta applicando 𝑊 a 𝐼:

dove (𝑥,𝑦) sono le coordinate dei pixel, e ​ e ​ sono le componenti della mappa di deformazione lungo gli assi x e y.

La mappa di deformazione è costruita in manira pseudo-randomica, utilizzando 2 parametri principali:

* **k**, che definisce la dimensione della pertubazione nell’immagine (figua 4);
* **s**, che definisce la quantità di perturbazione (figua 5):

  
Figura 4: Esempi di Wanet variando k con s=0.5

  
Figura 5: Esempi di WaNet variando s con k=4

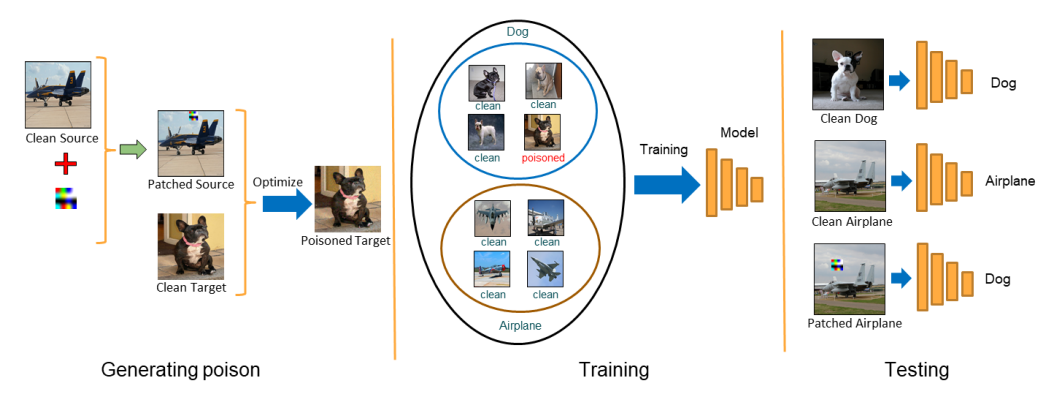
La mappa di deformazione è applicata a una frazione del dataset di training, distorcendo leggermente le immagini selezionate. Questa deformazione è progettata per essere sufficientemente lieve da non alterare la percezione visiva dell'immagine, ma abbastanza consistente da essere riconosciuta dal modello.

Dopo l'addestramento, il modello funzionerà normalmente su immagini pulite. Tuttavia, quando una nuova immagine viene distorta usando la stessa mappa di deformazione utilizzata durante l'addestramento, il modello classificherà erroneamente questa immagine come appartenente alla classe target.

# Hidden Trigger Backdoor

**Hidden trigger Backdoor** (**HTBD**) [7] è un attacco di tipo backdoor poisoning white box, in cui l'attaccante genera un insieme di immagini avvelenate che visivamente appaiono appartenere alla categoria target (figura 6), utilizzando un algoritmo PGD, mantenendo però segreto il trigger che verrà poi utilizzato effettuare la misclassification. Successivamente, queste immagini avvelenate vengono aggiunte al set di dati di addestramento con etichette apparentemente corrette, come nell’tecnica di Clean Label [5], corrispondenti alla categoria target. La vittima, ignara della presenza delle immagini avvelenate, utilizza questi dati per addestrare il modello di deep learning.

Quindi, durante la fase di test, l'attaccante introduce il trigger segreto nelle immagini della **categoria** **sorgente** per ingannare il modello e indurlo a classificare le immagini sorgenti nella **categoria** delle immagini **target** perturbate. A differenza di molti attacchi precedenti che utilizzano trigger visibili, in questo caso le immagini avvelenate non mostrano alcun trigger visibile e l'attaccante rivela il trigger solo al momento del test, quando è troppo tardi per implementare misure difensive.

  
Figura 6: Manipiulation datasets flow per HTBD

Più formalmente, data un’immagine target t, un’immagine sorgente s e un patch trigger p, viene applicato il trigger su s per ottenere l’immagine sorgente con il trigger denominata . Otteniamo così l’immagine ottimizzata z risolvendo il seguente problema di ottimizzazione:

dove f rappresenta le caratteristiche intermedie del modello (cioè i calcoli applicati dal modello per ottenere le features) e è un piccolo valore che assicura che l’immagine avvelenata z non sia visivamente distinguibile da t.

Per andare a risolvere il problema di ottimizzazione e trovare z viene applicato PGD. L’obbiettivo è quello di minimizzare la differenza tra le caratteristiche intermedie dell’immagine avvelenata z e quelle dell’immagine sorgente con la patch .

Si utilizza la discesa del gradiente per ridurre la funzione di perdita; in ogni passo iterativo dell’ottimizzazione si calcola il gradiente della funzione di Loss rispetto a z e si aggiorna z nella direzione che minimizza la perdita. Alla fine di ogni aggiornamento si proietta z per verificare che il vincolo sia rispettata e che quindi non si allontani troppo dall’immagine t. Generando così l’intero vettore di immagini perturbate.

1. Dataset

Il dataset scelto per questo studio comprende i **CT scan** (o TC, tomografia computerizzata[[5]](#footnote-5)) dei polmoni di individui infetti (figura 7) e non infetti (figura 6) da SARS-COVID-19.

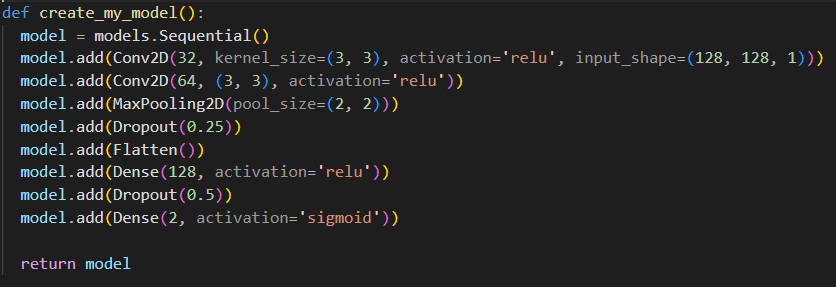
Dal punto di vista medico [2], è possibile determinare se un individuo è infetto da SARS-COVID-19 attraverso l'analisi delle immagini delle scansioni TC, poiché la malattia provoca specifici **segni di polmonite** (figura 8) visibili nelle immagini. Questi segni includono opacità a vetro smerigliato, consolidamenti polmonari e altre anomalie caratteristiche che possono essere identificati dai modelli di machine learning addestrati su tali dati.

|  |  |
| --- | --- |
|  |  |
| Figura 7: Polmoni di individuo infetti da SARS-COVID-19 | Figura 8: Polmoni di individuo non infetti da SARS-COVID-19 |

1. Metodi

# Rete neurale utilizzata

Per la sperimentazione è stata scelta una semplice rete neurale convoluzionale (CNN) (figura 9). Un tipo di rete neurale particolarmente efficace per elaborare dati con una struttura a griglia, come le immagini.

  
Figura 9: Codice che descrive la rete neurale utilizzata

## Spiegazione della rete neurale

Una rete neurale convoluzionale[[6]](#footnote-6) (CNN) è un tipo di rete neurale progettata per riconoscere e classificare pattern in dati strutturati come immagini. Le CNN utilizzano strati convoluzionali per estrarre feature locali dalle immagini, strati di pooling per ridurre le dimensioni spaziali delle feature map e strati densi per effettuare la classificazione finale. Le CNN sono particolarmente efficaci nell'elaborazione di immagini grazie alla loro capacità di catturare relazioni spaziali complesse attraverso l'applicazione di filtri convoluzionali che attraversano l'intera immagine.

1. **Conv2D Layers**:

* Conv2D(32, kernel\_size=(3, 3), activation='relu', input\_shape=(128, 128, 3)): Questo strato convoluzionale applica 32 filtri di convoluzione di dimensione 3x3 all'immagine di input, utilizzando la funzione di attivazione ReLU (Rectified Linear Unit). Definisce anche la shape dell'inpu, un'immagine di 128x128 pixel con 3 canali, cioè RGB.
* Conv2D(64, (3, 3), activation='relu'): Un altro strato convoluzionale che applica 64 filtri di convoluzione di dimensione 3x3, sempre con attivazione ReLU.

1. **MaxPooling2D**:

* MaxPooling2D(pool\_size=(2, 2)): Questo strato riduce le dimensioni spaziali delle feature map di un fattore di 2, prendendo il massimo valore in ogni finestra di 2x2 pixel. Questo aiuta a ridurre il numero di parametri e a controllare l'overfitting[[7]](#footnote-7).

1. **Dropout**:

* Dropout(0.25): Durante l'addestramento, questo strato spegne casualmente il 25% dei neuroni per evitare l'overfitting.
* Dropout(0.5): Un altro strato di dropout con il 50% di spegnimento dei neuroni, applicato prima del layer di output.

1. **Flatten**:

* Flatten(): Questo strato appiattisce le feature map 2D in un vettore 1D, rendendo i dati adatti per essere passati attraverso strati completamente connessi (dense layers).

1. **Dense** **Layers**:

* Dense(128, activation='relu'): Un strato denso con 128 neuroni e attivazione ReLU, che serve come strato intermedio per apprendere combinazioni complesse di feature.
* **Dense**(2, activation='sigmoid'): Lo strato di output con 2 neuroni e attivazione sigmoide[[8]](#footnote-8), adatto per la classificazione binaria (infetto o non infetto da SARS-COVID-19).

## Parchè lavorare sulla scala dei grigi?

Lavorare sulla scala dei grigi per analizzare le scansioni TC (tomografia computerizzata) con una rete neurale offre diversi vantaggi:

* **Riduzione della Complessità Computazionale**: Le immagini in scala di grigi hanno un solo canale di colore (intensità luminosa), mentre le immagini a colori RGB ne hanno tre (rosso, verde e blu). Lavorare con un solo canale riduce il numero di parametri nel modello e, di conseguenza, il tempo di addestramento e le risorse computazionali necessarie.
* **Informazioni Pertinenti**: Le scansioni TC sono generalmente rappresentate in scala di grigi perché contengono informazioni sufficienti per l'analisi medica. La variazione di intensità nei pixel di una scansione TC rappresenta le diverse densità dei tessuti corporei, fornendo dettagli cruciali per l'identificazione di anomalie come la polmonite da SARS-COVID-19.
* **Efficienza dei Modelli**: Le reti neurali convoluzionali (CNN) possono essere altrettanto efficaci nell'estrarre caratteristiche rilevanti da immagini in scala di grigi. Ridurre i dati a un singolo canale mantiene tutte le informazioni necessarie per l'analisi medica senza sacrificare la qualità diagnostica.
* **Minore Overfitting**: Con meno dati da elaborare (un solo canale invece di tre), il modello è meno propenso all'overfitting, ovvero a imparare rumori e dettagli non rilevanti, migliorando la generalizzazione su nuovi dati.

Infatti come sperimentato, il modello addestrato su immagini basate sulla scala dei grigi risulta avere un’accuratezza in fase validazione superiore di circa il 5% e una loss con meno oscillazioni e in un range più ristretto.

# Come verranno sperimentati gli attacchi

Nel presente studio verranno eseguite le tecniche descritte al fine di ottenere una misclassification della label non-COVID, quindi facendo in modo che le medesime immagini vengano classificate come facenti parte della classe COVID.

## Utilizzo Di FGSM

Nel presente studio il metodo di FGSM, verrà utilizzato non come metodo di evasion ma bensì come metodo di poisoning, vista la sua capacità di perturbare le immagini senza che si noti all’occhio umano, al fine di sperimentare se può essere utilizzato per degradare le performance della rete neurale colpita.

## Utilizzo di BadNets

BadNets verrà utilizzato come attacco di backdoor poisoning come descritto a Stato dell’Arte per effettuare una classificazione errata delle immagini non-COVID nella label COVID.

## Utilizzo di Clean Label

Nella seguente sperimentazione Clean Label verrà testato come descritto nello Stato dell’Arte in modo da effettuare la misclassification delle immagini non-COVID con il trigger patch applicato, come COVID.

## Utilizzo di WaNet

WaNet verrà utilizzato come attacco di backdoor poisoning come descritto a Stato dell’Arte per effettuare una classificazione errata delle immagini non-COVID nella label COVID. Tuttavia effettuando una modifica geometrica delle immagini potrebbe non funzionare, visto che le features che determinano la predizione nel presente caso non sono di tipo geometrico bensì dipendono dalla dimensione, pertanto questo attacco potrebbe non ingannare il modello.

## Utilizzo di Hidden Trigger Backdoor

Hidden Trigger Backdoor verrà utilizzato come metodo di backdoor poisoning white box, effettuando il poisoning delle immagini COVID utilizzando come sorgenti quelle non-COVID.

1. Sperimentazione

L’obiettivo del presente esperimento è valutare l’efficacia degli attacchi di poisoning e backdoor poisoning descritti, sul modello di classificazione di immagini presentato. Per effettuare i dovuti test è stato utilizzato il linguaggio di programmazione Python, il cui codice è disponibile nel seguente repository: https://github.com/nicolabalzano/CovidClassificationCNNAttack. Gli esperimenti sono stati condotti senza troppi ostacoli grazie alla libreria **Adversarial Robustness Toolbox** (**ART**), che fornisce strumenti avanzati per la creazione e la valutazione di attacchi AML, mentre la tecnica WaNet è stata implementata manualmente.

Nella sperimentazione è stato definito un modulo **util.py** che mette a disposizione delle funzioni utilizzate per rendere la sperimentazione più agevole, automatizzando operazioni ripetitive e semplificando l'implementazione degli esperimenti.



# Protocollo seguito

Il processo seguito nello sviluppo della sperimentazione per tutte le tecniche di AML è stato il seguente:

1. **Creazione del file CSV**:
   * Un file CSV è stato creato per associare il path del file di ogni immagine alla rispettiva classe di appartenenza, sia per il dataset di test che per quello di addestramento. Questo facilita l'organizzazione e l'accesso ai dati durante gli esperimenti.
2. **Recupero dei Datasets**:
   * I dataset sono stati caricati dalla memoria. Per l'addestramento, sono state utilizzate 1000 immagini per classe, di cui il 20% è stato riservato per la validazione. Le immagini rimanenti sono state utilizzate per il testing. Questo approccio garantisce un'adeguata separazione tra dati di addestramento, validazione e test.
3. **Addestramento e Validazione del Modello Originale**:
   * Il modello di classificazione è stato addestrato utilizzando il dataset di addestramento pulito. Durante l'addestramento, il modello è stato validato periodicamente per monitorare le sue prestazioni e prevenire l'overfitting.
4. **Testing del Modello Originale**:
   * Dopo l'addestramento, il modello è stato testato utilizzando il dataset di test pulito per valutare le sue prestazioni di base in assenza di attacchi.
5. **Selezione della Label da Avvelenare**:
   * È stata selezionata una specifica classe (ad esempio, COVID) le cui immagini verranno avvelenate. L'obiettivo è indurre una misclassification delle immagini appartenenti all'altra classe (ad esempio, non-COVID) come appartenenti alla classe target avvelenata. Utilizzando una percentuale di avvelenamento come valore minimo il 5% e come massimo il 50%, per categoria target.
6. **Poisoning delle Immagini**:
   * Le immagini della classe target sono state avvelenate secondo le specifiche di ogni attacco. Questo può includere l'aggiunta di perturbazioni minori o l'inserimento di trigger visibili per attacchi di backdoor.
7. **Riaddestramento e Validazione del Modello con Dataset Alterato**:
   * Il modello è stato riaddestrato utilizzando il dataset alterato contenente le immagini avvelenate. La validazione è stata condotta come in precedenza per monitorare le prestazioni del modello riaddestrato.
8. **Testing del Modello Riaddestrato**:
   * Il modello riaddestrato è stato testato utilizzando sia il dataset di test originale sia un dataset di test contenente le immagini con il trigger inserito, nel caso di attacchi di backdoor poisoning. Questo passaggio è cruciale per valutare l'efficacia degli attacchi in condizioni realistiche.
9. **Valutazione**:
   * Sono stati valutati i parametri del modello ottenuti tramite le predizioni, concentrandosi soprattutto sulla percentuale di accuratezza.

# Parametri di addestramento

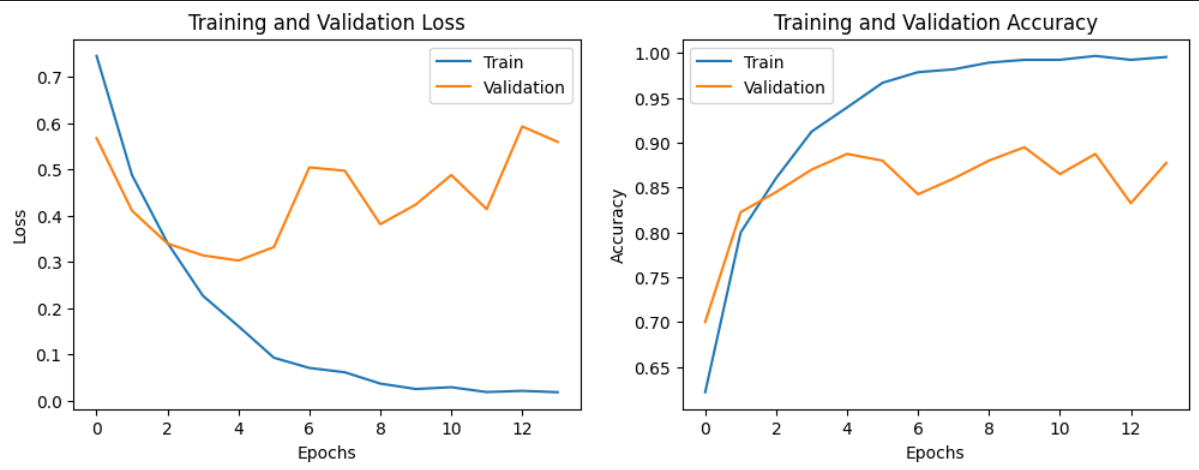
Per addestrare il modello sono state impostate un massimo di 25 epoche con una condizione di stop quando l’accuratezza in fase di training non saliva dopo 2 epoche.

1. Risultati

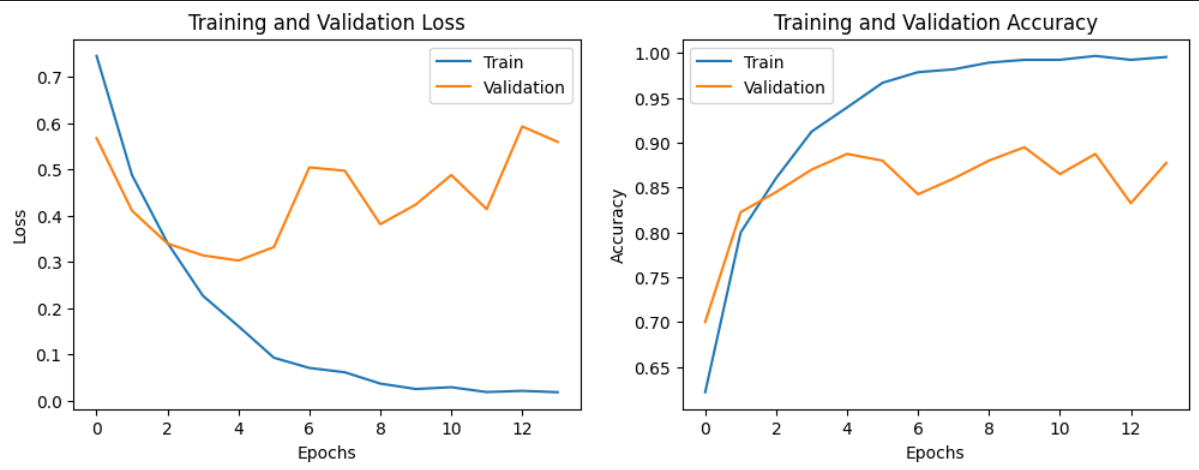
# Modello originale

Al fine di capire come viene degradato il modello tramite i vari attacchi, sono stati effettuati dei test per comprendere le capacità del modello originale.

Durante la fase di training (figura 10) il modello arriva ad un’accuratezza finale del 99,71% in 14 epoche, mentre in fase di valutazione l’accuratezza oscilla toccando anche l’89%, il che indica comunque che il modello non si trova in overfitting.

  
Figura 10: Andamento dell’accuratezza durante l’addestramento e la validazione

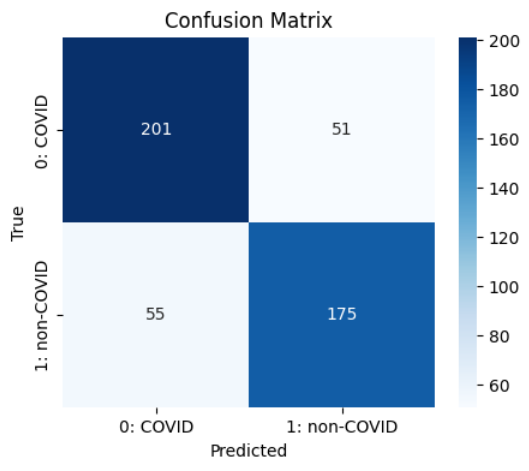
Per quanto riguarda la funzione di loss (figura 11) in questo caso una cross entropia binaria[[9]](#footnote-9), durante l’allenamento tende continuamente a scendere mentre durante la fase di validazione oscilla tra 0.4 e 0.6 il che sembra essere dovuto alla ridotta dimensione del dataset di validazione, nonché il 20% del dataset totale di 2000 immagini, questo potrebbe portare la loss a generare queste tipo di oscillazioni.

  
Figura 11: Andamento della funzione di Loss durante l’addestramento e la validazione

Il modello mostra prestazioni accettabili (tabella con una precisione, richiamo e F1-score medi del 78%. Tuttavia, c'è una leggera differenza nelle prestazioni tra le due classi, con la Classe 0 che ha metriche leggermente migliori rispetto alla classe 1. Questa differenza potrebbe essere dovuta alla distribuzione dei dati di addestramento o a una maggiore difficoltà del modello nel distinguere i campioni della classe 1. Globalmente, il modello sembra ben bilanciato e fornisce buone prestazioni generali nel compito di classificazione, come è possibile comunque vedere dalla matrice di confusione (figura 12).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Classification Report** | | | | |
|  | **Precision**[[10]](#footnote-10) | **Recall**[[11]](#footnote-11) | **F1-score**[[12]](#footnote-12) | **Support**[[13]](#footnote-13) |
| **COVID** | 0.79 | 0.80 | 0.79 | 252 |
| **non-COVID** | 0.77 | 0.76 | 0.77 | 230 |
| **Accurancy** |  |  | 0.78 | 482 |
| **Macro avg** | 0.78 | 0.78 | 0.78 | 482 |
| **Weighted avg** | 0.78 | 0.78 | 0.78 | 482 |

Tabella 1: Report della valutazione del modello

  
Figura 12: Matrice di confusione del modello originale

# FGSM

FGSM rivisitato come tecnica di poisoning ha permesso di ottenere dei risultati non soddisfacenti (tabella 2).

Lo scopo della seguente tecnica è stata quella di effettuare un misclassification generale di ogni classe, tuttavia ci si è concentrati sull’effettuare il poisoning delle immagini non-COVID, in modo da perturbare la classe scelta così che la predizione del modello ricada nella classe opposta.

## Test con = 0.01

Utilizzando come un valore di 0.01, dai risultati (tabella 2) si riscontra che la perturbazione inserita è troppo minima il che rende il test inefficace, infatti con una percentuale di avvelenamento del dataset target del 20%, si ottiene comunque un’accuratezza del 90% sulla predizione del dataset di test target.

## Test con = 0.05

Utilizzando come un valore di 0.05 e una percentuale di poisoning del 5% o 10%, non si ottengono risultati soddisfacenti, il modello continua predire in maniera abbastanza ottimale le immagini, perciò si è proceduto aumentando la percentuale di poisoning fino al 50%.

Tuttavi anche con una percentuale molto alta di avvelenamento il modello non sembra esserne particolarmente affetto.

## Test con = 0.1

Anche con una perturbazione uguale a 0.1 (tabella 2), questa tecnica non sembra essere molto utilizzabile per effettuare un attacco di poisoning.

## Test senza label target

Visto che non sembra essere possibile effettuare un attacco di poisoning utilizzando un label target, si è effettuato un test effettuando il poisoning delle immagini in maniera randomica, con una percentuale di dataset colpito pari al 50%.

Tuttavia anche con una = 0.1 senza effettuare il poisoning di una label specifica il modello sembra non effettuare una misclassification, le percentuali di accuratezza rimangono circa al 78%.

## Conclusione

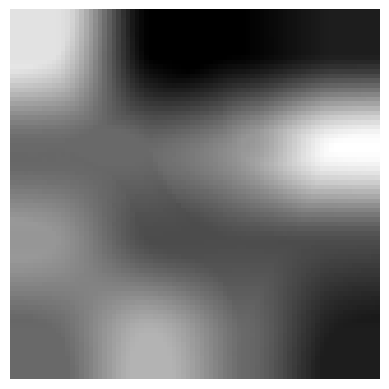
Dai dati ottenuti possiamo affermare che FGSM non può essere utilizzato come metodo di posioning poichè non sembra infliggere danni al modello, bensì se comunemente usato come descritto a Stato dell’arte, cioè come metodo di evasion, si riesce a far scendere l’accuratezza del modello, fino al 30% anche con una = 0.01.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **% avvelenata del dataset target** | **Accuratezza in fase di** | | | **Immagine perturbata** | **Immagine originale** |
| **training** | **test** | **test su label target** |
| 0.01 | 5% | 99.37% | 77.18% | 86.96% |  |  |
| 20% | 99.87% | 76.18% | 83.48% |
| 0.05 | 5% | 99.87% | 77.58% | 83.48% |  |
| 10% | 99.70% | 76.14% | 75.22% |
| 50% | 99.84% | 78.42% | 81.74% |
| 0.1 | 5% | 99.87% | 77.18% | 80% |  |
| 10% | 99.74% | 75.85% | 78.85% |

Tabella 2: Risultati FGSM come poisoning attack, target label non-COVID

# BadNets

È stato applicato l'attacco BadNets utilizzando la label COVID come classe target, con l'obiettivo di far classificare al modello le immagini non-COVID, contenenti il trigger (figura 13), come appartenenti alla classe COVID. I risultati ottenuti da vari test mostrano l'efficacia di questo attacco, evidenziando come il modello sia stato ingannato nel classificare erroneamente le immagini avvelenate, facendo scendere di molto il livello di accuratezza della rete neurale.

   
Figura 13: Backdoor trigger utilizzato

Effettuando differenti test, modificando la dimensione del trigger e la percentuale del dataset aggiunto per effettuare l’attacco, si sono riscontrati risultati ben differenti.

## Trigger 5x5

Il test con una dimensione costante del trigger di 5x5 px ripota che (tabella 3) con l’aumento della percentuale di dati avvelenati, l’accuratezza del modello sia in fase di training che in fase di test tende a diminuire. Tuttavia, ciò che è più rilevante è che l’accuratezza in fase di test con il trigger diminuisce drasticamente, dimostrando che l’attacco diventa più efficace con un maggior numero di dati avvelenati. Il compromesso qui è tra mantenere alte le accuratezze in fase di training e test per dati puliti, mentre si deduce l'accuratezza in presenza del trigger, ottenendo così un attacco di backdoor più subdolo ed efficiente.

Il **miglior** **risultato** (figura 14) ottenuto nel presente test è con una percentuale di avvelenamento pari al 20% che fa scendere l’accuratezza del modello fino al 30% anche se cala del 5% l’accuratezza in fase di training, il che rendere l’attacco più rilevabile.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| 5% | 10% | 20% | 40% |

Figura 14: Matrice di confusione del test con dimensione del trigger di 5x5

## Trigger 2x2

Utilizzando un trigger di dimensione 2x2 px, dai dati ottenuti (tabella 3) si può osservare che il trigger non è sufficientemente grande per essere considerate come una features, il rende quindi l’attacco inefficace utilizzando questo parametro.

## Trigger 7x7

Con un trigger di dimensione 7x7 px, è maggiormente rilevabile l’attacco anche ad occhio nudo, senza che si presti particolare attenzione. Tuttavia con se si usa percentuale di avvelenamento ridotta è difficile che venga rilevato, nel primo caso (tabella 3), con il solo 5% del dataset target avvelenato l’accuratezza del modello è scesa fino al 33.43%, quasi identica allo stesso risultato che si è ottenuto con un trigger di dimensione 5x5 e percentuale di avvelenamento pari al 40%. Possiamo quindi riaffermare che al crescere della dimensione del trigger patch aumenta l’efficacia dell’attacco, quindi la backdoor è più efficace.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 5% | 10% | 20% |

Figura 15: Matrice di confusione del test con dimensione del trigger di 7x7

## Conclusioni

Dai dati ottenuti il risultato migliore definito non solo come accuratezza ottenuta ma anche come difficoltà di rilevamento è stato quello con i medesimi parametri:

* dimensioni della patch = 5x5 px;
* percentuale di dataset avvelenato = 10%.

Questo tipo di attacco è però semplice da rilevare visto che usando le tecniche di **augmentation** e **label flipping** basta controllare la dimensione del dataset prima di utilizzarlo o fare un controllo manuale delle label con le relative immagini, o verificarlo tramite un’altra rete neurale protetta.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Dimensione del trigger** | **% avvelenata del dataset target** | **Accuratezza in fase di** | | | **Immagine perturbata** |
| **training** | **test** | **test con trigger** |
| 2x2 | 5% | 98.15% | 78.84% | 88.26% |  |
| 20% | 98% | 77.39% | 87.39% |
| 5x5 | 5% | 98.43% | 76.76% | 55.65% |  |
| 10% | 96.71% | 81.12% | 44.78% |
| 20% | 94.25% | 78.63% | 30% |
| 40% | 89.35% | 76.14% | 30.43% |
| 7x7 | 5% | 97.42% | 78.84% | 33.48% |  |
| 10% | 94.96% | 79.05% | 39.13% |
| 20% | 91% | 75% | 34.78% |

Tabella 3: Risultati di BadNets

# Clean Label

Clean label è stato applicato in modo da effettuare la stessa misclassification del metodo precedente, quindi il trigger patch è stato applicato sulle immagini della classe COVID in fase di training per poi applicarlo sulle immagini non-COVID in fase di test in modo da attivare la backdoor.

I seguenti test sono stati effettuati con una massima = 0.05, in modo da minimizzare il rumore inserito da PGD e renderlo meno rilevabile.

## Trigger 2x2

Con un trigger grande 2x2 px ci si aspettava come nel test dell’attacco BadNets che il trigger fosse troppo piccolo da essere considerato come una features, tuttavia visto l’utilizzo di un adversarial trainer che introduce rumore tramite l’algoritmo di PGD, l’immagine viene disturbata a tal punto che il modello considera il trigger patch come una features facendo scendere l’accuratezza del modello fino al 53% quando si infetta il 20% del dataset target (tabella 4) e si testa il modello con il trigger inserito sulle immagini della classe non target.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 5% | 10% | 20% |

Figura 16: Matrice di confusione del test con dimensione del trigger di 2x2

## Trigger 5x5

Con una dimensione di 5x5 px del trigger si ottengono risultati (tabella 4) equiparabili a quelli ottenuti nella tecnica precedentemente testata (tabella 3), con un lieve miglioramento su una percentuale di poisoning pari al 10% del dataset target e un miglioramento negativo dell’accuratezza del 5% con il 20% di poisoning.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 5% | 10% | 20% |

Figura 17: Matrice di confusione del test con dimensione del trigger di 5x5

## Trigger 7x7

Con una patch di dimensioni così elevata si ottengono risultati più che soddisfacenti (tabella 4) ma a discapito di un’alta probabilità di rilevamento.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 5% | 10% | 20% |

Figura 18: Matrice di confusione del test con dimensione del trigger di 7x7

## Conclusioni

In conclusione questa tecnica sembra essere molto efficace con il dataset in questione, e ad occhio umano è difficilmente rilevabile vista il ridotto rumore introdotto tramite i gradienti e la piccola dimensione del trigger di 2x2 px.

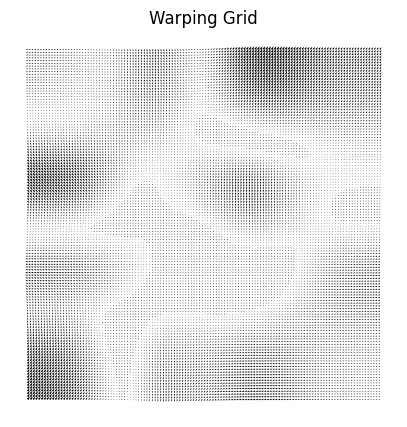
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Dimensione del trigger** | **% avvelenata del dataset target** | **Accuratezza in fase di** | | | **Immagine perturbata** |
| **training** | **test** | **test con trigger** |
| 2x2 | 5% | 99.81% | 81.20% | 60.59% |  |
| 10% | 99.87% | 80.68% | 57.20% |
| 20% | 99.42% | 79.88% | 53.04% |
| 5x5 | 5% | 99.67% | 79.69% | 42.52% |  |
| 10% | 99.81% | 79.88% | 40.80% |
| 20% | 98.98% | 80.88% | 35.25% |
| 7x7 | 5% | 98.81% | 80.88% | 32.79% |  |
| 10% | 98.43% | 81.12% | 33.02% |
| 20% | 97.89% | 79.98% | 32.02% |

Tabella 4: Risultati di Clean Label

# WaNet

Wanet è stato applicato in modo da effettuare la stessa misclassification dei metodi precedenti, perciò la deformazione geometriche è stata applicate sulle immagini della classe COVID in fase di training per poi applicarla sulle immagini non-COVID in fase di test in modo da attivare la backdoor.

Mentendo la grandezza della griglia costante a k=4 e variando la s si sono ottenuti differenti risultati.

  
Figura 19: Esempio di griglia di deformazione geometrica con s=9 e k=4

## s = 2

Con un’intensità di perturbazione così bassa anche, il modello non sembra esserne colpito da come si può notare dall’accuratezza riscontrata in fase di testing.

Tuttavia aumentando la percentuale di dataset avvelenato è possibile vedere come l’accuratezza del modello in fase di testing scenda di circa l’1%, ciò sembra indicare che con questo tipo di attacco all’aumentare della percentuale di posioning diminuisca l’accuratezza del modello.

Visto che i trigger inseriti sono permutazioni pseudo-randomiche è stato effettuato un altro test al fine di valutare quanto infligge il fatto randomico sull’attacco in questione. Come è possibile vedere dai risultati, nel caso preso attualmente in considerazione, sembra variare ma non in modo eccesivo, guadagnando un -1% di accuratezza.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 5% | 10% - 1 | 10% - 2 |

Figura 20: Matrice di confusione del test con s=2

## s = 5

Con una s pari a 5 il modello inizia a risentirne, l’accuratezza scende fino al 67.13% confermando l’ipotesi sopra descritta, l’attacco potrebbe già essere considerato abbastanza efficace, riuscendo a far diminuire l’accuratezza del modello del 10% rispetto al modello originale.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 5% | 10% - 1 | 10% - 2 |

Figura 21: Matrice di confusione del test con s=5

## s = 7

Con un parametro s di questo livello l’immagine risulta molto perturbata e perciò rende la tecnica rilevabile senza neanche troppa attenzione ad occhio nudo. Ciononostante l’attacco è comunque efficace, riesce a guadagnare un altro 10% in meno di accuratezza nella predizione.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 5% | 10% - 1 | 10% - 2 |

Figura 22: Matrice di confusione del test con s=7

## s = 9

Con una s=9 si riscontra un grande traguardo in termini di accuratezza minima raggiunta, anche se a discapito di un alta probabilità di rilevamento per via dell’elevata malformazione nell’immagine. Anche in questo caso sembra influire molto il fattore randomico, toccando anche un distacco nell’accuratezza del 13% a parità di parametri.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 5% | 10% - 1 | 10% - 2 |

Figura 23: Matrice di confusione del test con s=9

## Conclusioni

Come detto nella sezione precedente il presente modello per predire se una TAC è COVID o non-COVID la valuta non a livello di forma geometrica ma a livello di dimensione delle features presenti, (in questo caso macchie bianche), perciò questo tipo di attacco sembra funzionare ma solo con un parametro s abbastanza elevato (tabella 5). Inoltre essendo pseudo-randomica la generazione della deformazione non è l’attacco “ideale” per un dataset di questo tipo, dopo diversi test infatti con griglie differenti a parità di parametri si possono ottenere risultati anche molto distanti.

WaNet si dimostra essere non molto efficace come ci si era aspettato, per via delle features nelle immagini dell’attuale dataset, per questo è anche facilmente rilevabile anche ad occhio nudo questo tipo di attacco, tuttavia ha comunque degli effetti abbastanza elevati riuscendo a far scendere nel caso migliore l’accuratezza del modello al 36%. Un grande **vantaggio** però è che lascia totalmente invariata l’accuratezza in fase di addestramento, di validazione e di test, nonché i classici indicatori che vengono alterati quando si sta subendo un attacco backdoor poisoning.

  
Figura 24: Immagine orignale

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **s** | **k** | **% avvelenata del dataset target** | **Accuratezza in fase di** | | | **Immagine perturbata** |
| **training** | **test** | **test con il trigger** |
| 2 | 4 | 5% | 99.87% | 79.25% | 81.74% |  |
| 4 | 10% - 1 | 99.86% | 78.80% | 80.68% |
| 4 | 10% - 2 | 99.81% | 79.05% | 79.09% |
| 5 | 4 | 5% | 99.81% | 78.90% | 69.16% |  |
| 4 | 10% - 1 | 99.78% | 79.05% | 67.13% |
| 4 | 10% - 2 | 99.65% | 79.05% | 69.29% |
| 7 | 4 | 5% | 99.25% | 78.63% | 62.17% |  |
| 4 | 10% - 1 | 99.75% | 77.18% | 60.65% |
| 4 | 10% - 2 | 99.63% | 79.69% | 62.26% |
| 9 | 4 | 5% | 99.62% | 79.88% | 54.35% |  |
| 4 | 10% -1 | 99.81% | 81.54% | 36.96% |
| 4 | 10% -2 | 99.36% | 78.54% | 50.45% |

Tabella 5: Risultati di WaNet, dimensione della griglia fissa a 4 (descritta nell’attacco a Stato dell’arte come grandezza ottimale)

# Hidden Trigger Backdoor

Hidden Trigger Backdoor è stato applicato in modo da effettuare la stessa misclassification dei metodi precedenti, quindi come classe target è stata utilizzata la label COVID, mentre come classe sorgente quella non-COVID, in modo da perturbare la prima così che in fase di test venga applicato il patch sulle immagini non-COVID.

I seguenti test sono stati effettuati con una = 0.05, in modo da minimizzare il rumore inserito da PGD e renderlo meno rilevabile all’occhio umano.

## Trigger 2x2

Con trigger di così ridotte dimensioni l’attacco va a buon fine, facendo scende l’accuratezza del modello fino al 55% durante il test quando il trigger viene rivelato. I risultati ottenuti sono molto simili alla tecnica di Clean Label, nonché molto positivi.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 5% | 10% | 20% |

Figura 25: Matrice di confusione del test con dimensione del trigger di 2x2

## Trigger 5x5

Con una patch 5x5 px si registra un’accuratezza in fase di test in media del 42% il che significa che la tecnica riesce ad ingannare molto bene il modello, lasciando invariate le altre accuratezze.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 5% | 10% | 20% |

Figura 26: Matrice di confusione del test con dimensione del trigger di 5x5

## Trigger 7x7

Con un trigger di dimensioni così elevate si riesce ad ottenere un livello di accuratezza davvero molto basso, circa il 30%, anche se a discapito di un 1% di accuratezza in meno durante la fase di training.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 5% | 10% | 20% |

Figura 27: Matrice di confusione del test con dimensione del trigger di 7x7

## Conclusione

Hidden Trigger Backdoor si è dimostrato essere un buon attacco, riuscendo ad ingannare il modello in maniera molto efficace, il grande vantaggio di questo metodo è di poter usare un trigger davvero molto grande poiché non viene rilevato in fase di trainig ma solo durante la predizione finale. Tuttavia non bisogna esagerare con la dimensione di quest’ultimo poiché potrebbe far variare loss, val\_loss e val\_accurancy che invece rimangono stabili con i limiti testati, non lasciando intendere che l’attacco sia in corso.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Dimensione del trigger** | **% avvelenata del dataset target** | **Accuratezza in fase di** | | | **Immagine perturbata** |
| **training** | **test** | **test con trigger** |
| 2x2 | 5% | 99.71% | 82.20% | 58.59% |  |
| 10% | 99.25% | 81.68% | 55.20% |
| 20% | 99.36% | 79.92% | 55.24% |
| 5x5 | 5% | 99.79% | 79.69% | 45.95% |  |
| 10% | 99.81% | 79.88% | 42.98% |
| 20% | 98.92% | 80.82% | 41.99% |
| 7x7 | 5% | 98.90% | 78.92 % | 33.79% |  |
| 10% | 98.63% | 78.12% | 30.06% |
| 20% | 98.27% | 77.98% | 30.68% |

Tabella 6: Risultati di Hidden Trigger backdoor con masima pari a 0.05

# Considerazioni finali

Calcolando il **fooling rate** come , possiamo definire una funzione per valutare il tasso di successo di ogni tecnica testata, al fine di effettuare una valutazione finale (tabella 7), verranno prese solo le combinazioni dei parametri migliori ottenuti in ogni test (evidenziante in giallo).

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Predizioni sul dataset target** | | **Fooling rate %** |
| **errate** | **totali** |
| **FGSM poisoning** | n.d. | n.d. | n.d. |
| **BadNets** | 127 | 230 | 55.22% |
| **Clean Label** | 90 | 39.13% |
| **WaNet** | 90 | 39.13% |
| **Hidden Trigger Backdoor** | 152 | 66.08% |

Tabella 7: Tasso di successo di ogni attacco con i parametri migliori

Analizzando i dati forniti, emerge chiaramente che l'attacco Hidden Trigger Backdoor è il più efficace in termini di fooling rate, con una percentuale del 66.08%. Questo significa che oltre due terzi delle predizioni sono errate, un dato che testimonia la sua elevata capacità di ingannare il modello. Inoltre, è il meno rilevabile tra tutti gli attacchi considerati, rendendolo particolarmente insidioso.

BadNets, sebbene abbia un fooling rate del 55.22%, risulta essere molto rilevabile. Questo suggerisce che, nonostante la sua efficacia nel compromettere il modello, è più facilmente individuabile rispetto ad altri metodi. La stessa osservazione vale per WaNet, che condivide lo stesso fooling rate del 39.13% con Clean Label, ma, nel nostro caso, è considerato rilevabile.

Clean Label, pur avendo un fooling rate identico a WaNet, è classificato come meno rilevabile. Questo implica che, sebbene meno efficace nel complesso rispetto a Hidden Trigger Backdoor, Clean Label riesce comunque a mantenere un buon livello di inganno riducendo le probabilità di essere scoperto.

In conclusione, **Hidden Trigger Backdoor** emerge come l'attacco **più subdolo** ed efficace, grazie al suo alto fooling rate e alla sua scarsa rilevabilità. Al contrario, BadNets, pur essendo efficiente nella compromissione, è facilmente rilevabile, il che ne limita l'utilizzo in contesti dove la furtività è cruciale. Clean Label e WaNet si posizionano in una fascia intermedia, bilanciando efficacia e rilevabilità in modo diverso a seconda delle specifiche necessità del contesto di applicazione.

1. Bibliografia

[1] Apostol Vassilev, Alina Oprea, Alie Fordyce, Hyrum Anderson (Gennaio 2024)  
Adversarial Machine Learning a Taxonomy and Terminology of Attacks and Mitigations

[2] igeasantimo blog (23 Marzo, 2023).   
https://www.igeasantimo.com/blog/il-ruolo-della-tc-nella-malattia-da-covid-

[3] Alexey Kurakin, Ian Goodfellow, and Samy Bengio (8 Luglio, 2016)  
Adversarial Examples in the Physical World

[4] Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg (22 agosto 2017)  
BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain

[5] Alexander Turner, Dimitris Tsipras, Aleksander Mądry (28 Settembre, 2018)  
Clean-Label Backdoor Attacks

[6] Anh Tuan Nguyen, Anh Tuan Tran (4 Marzo, 2021)  
WaNet – Imperceptible warping-based back-door attack

[7] Aniruddha Saha, Akshayvarun Subramanya, Hamed Pirsiavash (30 Settembre, 2019)  
Hidden Trigger Backdoor Attacks

1. Colui o coloro che hanno l'intento e le capacità di compromettere la sicurezza di un sistema o di un'organizzazione. [↑](#footnote-ref-1)
2. Tecnica di difesa contro gli attacchi avversari in cui il modello di machine learning viene addestrato utilizzando sia dati originali che dati deliberatamente perturbati [↑](#footnote-ref-2)
3. Solitamente il trigger è un’immagine di dimensione ridotte che viene applicata al di sopra di quella originale [↑](#footnote-ref-3)
4. Una misura di quanto le previsioni di un modello differiscono dai valori attesi [↑](#footnote-ref-4)
5. Comunemente nota come TAC [↑](#footnote-ref-5)
6. La convoluzione è un'operazione che combina due funzioni per produrre una terza funzione, nel ambiente di questo studio è utilizzata per estrarre caratteristiche (feature) dalle immagini. [↑](#footnote-ref-6)
7. Quando un modello di machine learning si adatta troppo strettamente ai dati di addestramento, catturando anche il rumore e le anomalie, e quindi non generalizza bene su dati nuovi o non visti, risultando in una scarsa performance predittiva. [↑](#footnote-ref-7)
8. Una funzione di attivazione utilizzata nelle reti neurali che mappa i valori di input in un intervallo compreso tra 0 e 1, seguendo una curva a forma di "S". Spesso utilizzata per problemi di classificazione binaria, come quello preso in studio. [↑](#footnote-ref-8)
9. Funzione di perdita comunemente utilizzata nei problemi di classificazione binaria [↑](#footnote-ref-9)
10. Percentuale di predizioni corrette, calcolata come (Veri Positivi / (Veri positivi + Falsi positivi) [↑](#footnote-ref-10)
11. Percentuale di quanti dei campioni positivi nel dataset sono stati correttamente identificati dal modello, calcolata come (Veri Positivi / (Veri positivi + Falsi negativi) [↑](#footnote-ref-11)
12. La media armonica della precisione e del richiamo [↑](#footnote-ref-12)
13. Numero di occorrenze per classe [↑](#footnote-ref-13)